Guida HTML5 di base

Il linguaggio HTML

Le pagine di Internet sono redatte con un linguaggio studiato per la gestione degli ipertesti: non si tratta di un linguaggio di programmazione in quanto non viene utilizzato per eseguire elaborazioni, ma di un **linguaggio di formattazione della pagina**.

Un ipertesto è un insieme di documenti che ha la possibilità di essere consultato in modo non sequenziale, passando da un documento all'altro attraverso collegamenti (link) realizzati mediante parole o immagini.

Esso può contenere elementi non solo testuali, ma anche immagini, suoni, filmati e qualunque tipo di oggetto trattabile da un sistema informatico. In ambiente Internet, in particolare utilizzando i riferimenti **URL** (***Uniform Resource Locator***), cioè gli indirizzi dei siti, si può accedere da un documento ad un altro registrato su un computer ubicato in qualsiasi parte del mondo, realizzando così un'attività che viene comunemente definita **navigazione** nella rete (*netsurfing*).

Il linguaggio **HTML** (***HyperText Markup Language***, cioè linguaggio che utilizza dei contrassegni per la formazione di ipertesti) consente di sfruttare i vantaggi e le prestazioni derivanti dall'organizzazione ipertestuale e dall'uso degli oggetti multimediali: questo linguaggio è diventato lo standard nell'architettura **WWW** (***World Wide Web***) per creare e riconoscere i documenti ipermediali.

Le specifiche del linguaggio sono stabilite dal W3C (***World Wide Web Consortium***). Il W3C è un'organizzazione non governativa internazionale (cioè un'organizzazione indipendente dal governo e dalla sua politica senza fini di lucro) che sovraintende alla definizione degli standard, delle specifiche, dei protocolli e delle linee guida che riguardano il World Wide Web (**www.w3.org** oppure **www.w3c.it** per la versione il lingua italiana).

Fondato nell'ottobre del 1994 al MIT (Massachusetts Institute of Technology) dal "padre" del Web, Tim Berners-Lee, in collaborazione con il CERN (il laboratorio dal quale Berners-Lee proveniva), oggi conta tra le sue fila più di 300 membri provenienti da tutto il mondo, tra cui Adobe, Apple, CERN, Google, IBM, Mozilla Foundation, Microsoft, Opera Software, Oracle, alcune università e un nutrito numero di produttori di computer e dispositivi mobili.

Le esigenze di inserimento di informazioni sempre più sofisticate e la necessità di rendere interattive con l'utente le pagine stesse, hanno introdotto via via numerose modifiche e ampliamenti al linguaggio stesso, creando versioni diverse.

Storia e cronologia di HTML

Il lungo cammino verso HTML5 parte dalla prima definizione di HTML avvenuta all'inizio degli anni '90. Il neonato **Internet** aveva bisogno di un linguaggio di formattazione dei contenuti che li rendesse fruibili in modo organizzato, questi sarebbero poi stati veicolati all'utente avvalendosi di un browser, un software cioè che li avrebbe mostrati così come indicato dai comandi (**tags**) che componevano il documento (**ipertesto**) HTML assieme all'informazione pura.

Tim Berners-Lee, padre del Web, lo definì basandosi sul **metalinguaggio SGML** (**Standard Generalized Markup Language**) e gli affiancò un protocollo di trasporto, il ben noto **HTTP**.

Che cos'è un metalinguaggio

Nella logica e nella teoria dei linguaggi formali per **metalinguaggio** si intende un linguaggio formalmente definito che ha come scopo la definizione di altri linguaggi artificiali che diciamo **linguaggi obiettivo** o **linguaggi oggetto** (nell'ambito di **SGML** e di **XML** si usa anche il termine **applicazioni**).

Il metalinguaggio SGML

L'SGML è un metalinguaggio definito come standard ISO (ISO 8879:1986 SGML) avente lo scopo di definire linguaggi da utilizzare per la stesura di testi destinati ad essere trasmessi ed archiviati con strumenti informatici, ossia per la stesura di documenti in forma leggibile da computer (*machine readable form*).

Principale funzione di SGML è la stesura di testi chiamati **Document Type Definition**, in acronimo **DTD**, ciascuno dei quali definisce in modo rigoroso la struttura logica che devono avere i documenti di un determinato tipo; si dice che questi documenti rispetto a SGML costituiscono un linguaggio obiettivo, ovvero una **applicazione**.

Il metalinguaggio XML

XML (**eXtensible Markup Language**) è un linguaggio di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

Costituisce il tentativo di produrre una versione semplificata di **SGML** che consenta di definire in modo semplice nuovi linguaggi di markup da usare in ambito web. Il nome indica quindi che si tratta di un linguaggio marcatore (*markup language*) estensibile (*eXtensible*) in quanto permette di creare tag personalizzati.

Rispetto all'HTML, l'XML ha uno scopo ben diverso: mentre il primo definisce una grammatica per la descrizione e la formattazione di pagine web (layout) e, in generale, di ipertesti, il secondo è un metalinguaggio utilizzato per creare nuovi linguaggi, atti a descrivere **documenti strutturati**. Mentre l'HTML ha un insieme ben definito e ristretto di tag, con l'XML è invece possibile definirne di propri a seconda delle esigenze.

Da HTML Tags ad HTML 4.01

Negli anni successivi l'affermazione del Web con la sua repentina ed inarrestabile diffusione aveva creato la necessità di estendere e migliorare il linguaggio originale (HTML Tags), inizialmente composto da una manciata di elementi. Dopo alcune revisioni avvenute tra il 1991 ed il 1992, nel 1993 Lee insieme allo **IETF** (Internet Engineering Task Force, una comunità di tecnici interessata allo sviluppo di Internet) pubblicò **"Hypertext Markup Language (HTML)" Internet-Draft**, il primo documento ufficiale che proponeva (*proposal*) una bozza avente lo scopo di formalizzare il linguaggio.

Dopo ulteriori revisioni, il lavoro di stesura delle specifiche sfociò in **HTML 2.0**, a metà del 1993.

Nel 1995 lo IETF rese pubblico un nuovo proposal che introduceva **HTML 3.0**. La specifica non ebbe successo per vari motivi, non ultimo dei quali l'imperversante **guerra dei browser** che in quel momento storico intercorreva tra **Netscape e Microsoft**.

Le due aziende decisero di implementare solamente un *subset* delle funzionalità descritte nelle 150 pagine del documento aggiungendo al contempo estensioni proprietarie che miravano soprattutto al controllo dello stile e del *look&feel* delle pagine web (**visual markup**).

Questo approccio era in aperto contrasto con la filosofia accademico/tecnica che intendeva HTML come un linguaggio esclusivamente di formattazione (**markup**).

Con l'abbandono dello IETF, formalizzato con la chiusura del proprio HTML Working Group nel 1996, la prima *Recommendation* del W3C uscì nel 1997 (**HTML 3.2**). Il documento si proponeva di ridurre la distanza tra le estensioni proprietarie promuovendone una sintesi accettabile ed adottando in parte i tags "stilistici" di Netscape.

Nel dicembre 1997 il W3C pubblicò una nuova Recommendation: **HTML 4.0** (nome in codice "Cougar") che prevedeva tre varianti:

- **Strict**: gli elementi deprecati erano proibiti.
- **Transitional**: gli elementi deprecati erano permessi.
- **Frameset**: in buona sostanza veniva permesso l'utilizzo dei soli elementi strettamente legati ai frames.

HTML 4.0 inoltre deprecava i tags Netscape relativi allo stile, tra i quali il tag **font**, caldeggiando in alternativa l'uso dei **CSS** (Fogli di stile – Cascading Style Sheets).

Preceduta e seguita da alcune piccole correzioni (**errata**), nel Dicembre del 1999 fu pubblicata la Recommendation **HTML 4.01** che attualmente rimane l'ultima rilasciata per il linguaggio.

XHTML

XHTML (eXtensible HyperText Markup Language) è una **applicazione di XML così come HTML lo è di SGML**. Il W3C fornisce questa definizione per la prima versione del linguaggio:

XHTML 1.0 è una riformulazione di HTML 4.01 in XML, combina la robustezza di HTML4 con la potenza di XML.

Il documento XHTML può essere validato da un *parser* XML, il documento HTML al contrario ne richiede uno specifico.

Qualche caratteristica:

- XML è case-sensitive per elementi ed attributi.
- il processamento di un documento che produca errori di parsing semplicemente viene terminato.
- il doctype deve essere sempre presente perché il documento possa essere validato.
- tutti i tags devono essere chiusi.

XHTML è quindi molto più restrittivo di HTML, in conseguenza del fatto che XML di cui è applicazione è un subset molto più restrittivo di SGML rispetto a quanto lo sia HTML.

Tra gli obiettivi della sua creazione la volontà di spingere gli autori di contenuti Web verso una sintassi più vicina all'XML allo scopo di semplificare **l'interoperabilità con altri formati XML come ad esempio SVG** (Scalable Vector Graphics).

La versione 1.0 diventò Recommendation W3C il 26 Gennaio 2000.

La spinta successiva soffiò nella direzione della **modularizzazione XHTML**, ossia la pacchettizzare del linguaggio in moduli con lo scopo di renderlo estensibile. L'idea era quella di renderlo flessibile per l'utilizzo su *media* come dispositivi *mobile* e TV connesse alla rete.

Il lavoro portò alla realizzazione di una nuova specifica: **XHTML 1.1**.

Oltre ad eliminare alcuni attributi ed aggiungere dei tags per migliorare il supporto per le lingue asiatiche il linguaggio **doveva essere trasmesso con un** *media type* **particolare (application/xhtml+xml) e non come HTML**; questa limitazione nè limitò fortemente l'adozione tanto da spingere nel 2009 il W3C a rilassare la restrizione permettendo che il documento si potesse servire come HTML.

Venne iniziato poi il lavoro di stesura delle specifiche di **XHTML 1.2** abbandonato definitivamente nel 2010.

XHTML 2.0

Nel 2002 il W3C rilasciò il primo *draft* per un nuovo linguaggio che non prevedeva la retrocompatibilità nè con XHTML 1.x nè con HTML 4.01.

XHTML 2.0 doveva adottare XML DOM al posto del Document Object Model, tra le novità prevedeva che ogni elemento avrebbe potuto comportarsi come un link a fronte della sola aggiunta di un attributo href e includeva tra gli altri un nuovo tag *nl* per implementare liste di navigazione.

Le pressioni esercitate in senso contrario al linguaggio dal WHATWG, dovute in massima parte al disaccordo sulla mancanza di *backward compatibility*, portarono all'abbandono definitivo del progetto nel 2009 spianando definitivamente la strada ad HTML5.

HTML5

Attualmente **HTML5** non è ancora uno standard rilasciato in modo completo. Infatti le sue specifiche sono ancora in fase di definizione e il loro definitivo rilascio è previsto per il 2014.

Nel frattempo, il W3C sta elaborando una serie di bozze delle specifiche allo scopo di delineare le caratteristiche della nuova versione del linguaggio, fino alla sua definizione finale.

Nel seguito dei capitoli vengono presentati gli <u>**elementi compatibili soltanto con la versione HTML5**</u>. Per visualizzare le pagine web contenenti elementi propri dell’ultima versione del linguaggio HTML è preferibile usare **Google Chrome** che fra tutti i browser presenti attualmente sul mercato è quello che fornisce un supporto molto esteso alle funzionalità di HTML5.

I documenti HTML

I documenti WWW, tipicamente scritti in HTML, sono dei normali testi di caratteri, e di conseguenza sono visibili e modificabili con qualunque programma di trattamento dei testi (ad esempio Blocco Note).

Un documento in formato Web può essere aperto con un browser. Sullo schermo viene visualizzata una pagina in formato grafico: la visualizzazione è il risultato di un'elaborazione del browser che interpreta i codici contenuti nel file e li trasforma in comandi per la costruzione della pagina.

I testi scritti in HTML si distinguono dai normali file **txt** o comunque da file di testo non HTML tramite l'estensione **.html** nel nome del file (o più semplicemente il suffisso abbreviato **.htm**).

Il suffisso non è un elemento obbligatorio, ma è utilizzato normalmente per identificare il tipo di file. In ambiente Windows i documenti in HTML vengono rappresentati con un'icona che raffigura il logo del browser predefinito nel computer dell'utente.

Tali documenti, quindi, non sono altro che dei **normali file di caratteri ASCII**; in più vengono aggiunti i **codici di formattazione della pagina** che prendono il nome di **tag** (cioè contrassegno, da cui deriva il termine *MarkUp* della sigla HTML) consistenti in sequenze di caratteri proprie del linguaggio HTML che non fanno parte del testo normale e che consentono di realizzare gli elementi caratteristici dell'ipertesto: i link (collegamenti ipertestuali), gli oggetti grafici e multimediali. Altri codici servono a semplificare l'impaginazione dei documenti (stili del testo, titolo dei documenti, paragrafi, liste), così da rendere il linguaggio utile anche per creare documenti complessi.

I tag di HTML: come scriverli

Struttura di un tag

Abbiamo detto che all'interno di ogni pagina è presente una serie di marcatori (i **tag**), a cui viene affidata la visualizzazione e che hanno differenti nomi a seconda della loro funzione. I tag vanno inseriti tra parentesi angolari (**<tag>**), la chiusura del tag viene indicata con una "/" (è il simbolo comunemente detto "slash". Quindi: **</tag>**). Il contenuto va inserito tra l'apertura e la chiusura del tag medesimo, secondo questa forma:

```
<tag attributi>contenuto</tag>
```

Ecco un esempio, con una sintassi che serve a disporre un testo giustificato a destra:

```
<p align="right">testo</p>
```

dall'esempio è evidente che la struttura di un attributo è: attributo="valore". Quindi in definita la struttura di un tag sarà:

```
<tag attributo_1="valore1" attributo_2="valore2">contenuto</tag>
```

Alcuni particolari tag non hanno contenuto – perché ad esempio indicano la posizione di alcuni elementi (come il tag delle immagini) –; conseguentemente questi tag non hanno neanche chiusura. La loro forma sarà dunque:

```
<tag attributi>
```

Ecco un esempio di markup per inserire un'immagine:

```
<img widht="20" height="20" src="miaImmagine.gif" alt="alt" />
```

Come si vede il tag non viene chiuso. Questo tipo di tag viene detto "empty", cioè "vuoto".

Le tipologie di tag

I tag HTML possono rappresentare oggetti (come ad esempio le immagini) o servire a suddividere la pagina in aree (come i 'div'). Ci sono diverse tipologie di tag e conoscerle diventa determinante per usare il tag giusto al posto giusto e per applicare in seguito le regole CSS.

Annidamento e indentazione

Una caratteristica importante del codice HTML è che i tag possono essere annidati l'uno dentro l'altro. Anzi molto spesso è necessario farlo.

Ad esempio:

```
<tag1 attributi>
   contenuto 1
   <tag2>contenuto 2</tag2>
</tag1>
```

Potremmo quindi avere ad esempio:

```
<div align="right">
   testo 1
   <p align="left">testo 2</p>
</div>
```

L'annidamento ci permette quindi di attribuire formattazioni successive al testo che stiamo inserendo.

Come si può vedere già nell'esempio, è una buona norma utilizzare dei **caratteri di tabulazione** (il tasto **Tab** a sinistra della lettera **Q**) per far rientrare il testo ogni volta che ci troviamo in presenza di un annidamento e man mano che entriamo più in profondità nel documento.

In pratica apertura e chiusura del tag si trovano allo stesso livello, mentre il contenuto viene spostato verso destra di un tab: non si tratta soltanto di un fattore visivo, ma l'allineamento di apertura e chiusura tag viene mantenuto anche se scorriamo in verticale il documento con il cursore.

Questa procedura si chiama **indentazione**, e grazie ad essa il codice HTML risulta più leggibile. Si confronti ad esempio:

```
<div align="right">testo 1<p align="left"> testo 2 </p></div>
```

con:

```
<div align="right">
   testo 1
   <p align="left">
      testo 2
   </p>
</div>
```

per il browser i due esempi sono equivalenti, ma per l'utente umano è evidente che la differenza è notevole: pensate ad una pagina complessa visualizzata in un unico blocco di testo: sarebbe del tutto illeggibile!

La sintassi di HTML5

Prima di scendere nei dettagli presentando i nuovi elementi e le nuove API definite nella specifica, è necessario spendere qualche momento per parlare delle regole sintattiche introdotte dall'HTML5, per larga misura ereditate e razionalizzate dalla precedente versione delle specifiche. In primo luogo familiarizziamo con il concetto noto di tag. Esistono tre distinte versioni di questa particella, ognuna di esse si applica selettivamente solo ad alcuni elementi:

```
Tag 'classico'
<p> bla bla bla bla ... </p>

Tag 'vuoto'
<img src="immagine_interessante.jpg" alt="Una immagine interessante">

Tag 'autochiudente'
<rect x="150" y="40" width="60" height="30" fill="black"
stroke="black"/>
```

Gli elementi HTML5 si possono dividere in tre categorie sulla base della tipologia di tag da usare per implementarli.

1. **Elementi normali**: sono quelli che possono racchiudere dei contenuti sotto forma di testo, commenti HTML, altri elementi HTML, etc. Sono elementi normali, dunque, i paragrafi (`<p>`), le liste (`<ul>`), i titoli (`<h1>`), etc. Salvo specifici casi, cui accenneremo nel seguito della lezione, gli elementi normali vengono definiti attraverso un **tag di apertura** (`<p>`) e un **tag di chiusura** (`</p>`).
2. **Elementi vuoti**: gli elementi vuoti sono quelli che non possono avere alcun contenuto. Per questi elementi si utilizza un tag 'vuoto'. Essi sono: `area`, `base`, `br`, `col`, `command`, `embed`, `hr`, `img`, `input`, `keygen`, `link`, `meta`, `param`, `source`, `track`, `wbr`.

Per gli elementi vuoti, la chiusura del tag, **obbligatoria** in XHTML, è invece **opzionale**. Possiamo dunque definire un tag <img> secondo le regole XHTML:

```
<img src="immagine.png" alt="testo" />
```

o seguendo la vecchie regole di HTML 4:

```
<img src="immagine.png" alt="testo">
```

Maiuscolo, minuscolo

Una delle differenze principali rispetto alle regole XHTML riguarda l'uso del maiuscolo e del minuscolo per definire un tag. In XHTML è obbligatorio usare il minuscolo. In HTML5 è consentito scrivere un tag usando anche il maiuscolo:

```
<IMG src="immagine.png" alt="testo">
```

Casi particolari

Esistono alcune casistiche per le quali un tag classico può mancare della sua particella di apertura o di chiusura; questo succede quando il browser è comunque in grado di determinare i limiti di operatività dell'elemento. Gli esempi più eclatanti riguardano i tag 'contenitori', come head, body e html, che possono essere omessi in toto a meno che non contengano un commento o del testo come istruzione successiva. È quindi sintatticamente corretto scrivere un documento come segue:

```
<meta charset="utf-8">
<title>Pagina HTML5 Valida</title>
<p>Un paragrafo può non avere la particella di chiusura
<ol>
   <li>e anche un elemento di lista
</ol>
```

Notiamo che, come mostrato nell'esempio, anche i tag `p` e `li` possono essere scritti omettendo la particella di chiusura, a patto che l'elemento successivo sia all'interno di una cerchia di elementi definita dalle specifiche. A fronte di queste opzioni di semplificazione è però errato pensare che la pagina generata dal codice di cui sopra manchi, ad esempio, dell'elemento `html`; esso è infatti dichiarato implicitamente ed inserito a runtime dallo user-agent.

Attributi

Anche rispetto alle definizione degli attributi HTML5 consente una libertà maggiore rispetto a XHTML, segnando di fatto un ritorno alla filosofia di HTML 4. In sintesi: **non è più obbligatorio racchiudere i valori degli attributi tra virgolette**. I casi previsti nella specifica sono 4.

Attributi 'vuoti': non è necessario definire un valore per l'attributo, basta il nome, il valore si ricava implicitamente dalla stringa vuota. Un caso da manuale:

Secondo le regole XHTML:

```
<input checked="checked" />
```

In HTML5:

```
<input checked>
```

Attributi senza virgolette: è perfettamente lecito in HTML5 definire un attributo senza racchiudere il valore tra virgolette. Esempio:

```
<div class=testata>
```

Attributi con apostrofo: il valore di un attributo può essere racchiuso tra due apostrofi (termine più corretto rispetto a 'virgoletta singola'). Esempio:

```
<div class='testata'>
```

Attributi con virgolette: per concludere, è possibile usare la sintassi che prevede l'uso delle virgolette per racchiudere il valore di un attributo. Il codice:

```
<div class="testata">
```

Semplificazioni

In direzione della semplificazione vanno anche altre indicazioni. Ci soffermiamo su quelle riguardanti due elementi fondamentali come `style` e `script`. La sintassi tipica di XHTML prevede la specificazione di attributi accessori come `type`:

```
<style type="text/css"> regole CSS... </style>
<script type="text/javascript" src="script.js"> </script>
```

In HTML5 possiamo farne a meno, scrivendo dunque:

```
<style> regole CSS... </style>
<script src="script.js"> </script>
```

Conclusioni

Come abbiamo sperimentato, la sintassi HTML5 si caratterizza per una spiccata **flessibilità e semplicità** di implementazione. Le osservazioni che abbiamo snocciolato in questa lezione sono chiaramente valide a patto di implementare la serializzazione HTML; ricordiamo infatti che le specifiche prevedono anche l'utilizzo di una sintassi XML attraverso l'uso delle istruzioni:

```
<!doctype html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

Infine, per una migliore leggibilità del codice sorgente, consigliamo di ricorrere il meno possibile all'utilizzo di elementi impliciti, scrivendo sempre tutti i tag necessari nella loro forma completa.

Gli attributi globali

Sono gli attributi che **si possono applicare a tutti gli elementi HTML**. Vediamo i principali, tralasciando i nuovi introdotti con HTML5:

Tag	Descrizione
accesskey	Specifica una combinazione di tasti per attivare il focus di un elemento
class	Specifica una o o più classi per un determinato elemento
dir	Specifica la direzione (in senso orizzontale) del contenuto di un elemento. I **valori** dell'attributo sono: **ltr**: da sinistra verso destra (default); **rtl**: da destra verso sinistra; **auto**: impostato dal browser in base al contenuto della pagina. `<p dir="rtl">Write this text right-to-left!</p>`
id	Specifica un identificativo univoco per l'elemento (utile per manipolare l'elemento all'interno di una procedura Javascript o identificarlo in un foglio di stile)
lang	Specifica il codice della lingua utilizzata per il contenuto di un elemento
style	Specifica uno stile "inline" di un elemento
tabindex	Specifica l'ordine con il quale viene evidenziato un elemento, quando viene usato il tasto "Tab" per la consultazione delle pagine web
title	Specifica un'informazione aggiuntiva relativa ad un elemento. Esempio: `<p title="Free Web tutorials">W3Schools.com</p>`

Tipologie di tag

Un altro concetto importante è che gli elementi vengono classificati nella trattazione a fogli di stile secondo tre tipologie:

Elementi di blocco	Sono sostanzialmente gli elementi che costituiscono un blocco attorno a sé, e che di conseguenza vanno a capo, come i paragrafi, le tabelle, le form, ma soprattutto i **div**
Elementi "inline"	Sono gli elementi che – non andando a capo – possono essere integrati nel testo, come i collegamenti o le immagini oppure gli **span**
Liste	Liste numerate o non numerate

Anatomia di una pagina HTML

Il codice HTML si caratterizza sempre per la presenza al suo interno di tre tag fondamentali:

<html>
<head>
<body>

La struttura di base di ogni documento HTML è quindi articolata in questo modo:

```
<html>
   <head>
      [...]
   </head>
   <body>
      [...]
   </body>
</html>
```

Il browser sa che deve leggere tutto ciò che è contenuto entro i tag **<html>**…**</html>** come codice HTML ed è in grado di riconoscere il punto di inizio e quello di chiusura rispettivamente della testa e del corpo del documento.

Un documento HTML si divide in due parti fondamentali: l'intestazione e il corpo del documento.

<head> = intestazione

Gli elementi **<head>** e **</head>** sono posti immediatamente dopo l'apertura del tag **<html>** e racchiudono l'intestazione vera e propria del documento.

In questa sezione sono presenti tutte le informazioni necessarie al browser per una corretta interpretazione del documento, ma che l'utente non visualizza sulla schermo, a meno che non apra la finestra del browser che fa visualizzare il codice sorgente.

L'intestazione contiene: il **titolo della pagina**, i **fogli di stile** (incorporati o collegati), **script Javascript** e i **meta-tag** (alcuni dei quali, le parole chiave ad uso esclusivo dei motori di ricerca). Le parole chiave ("keywords") sono delle informazioni che vengono passate al browser tramite dei tag specifici e che servono ai motori di ricerca per comprendere il contenuto del sito.

```
<html>
<head>
   <title>Title of the document</title>
   <link rel="stylesheet" type="text/css" href="css/stile.css">
   <meta name="keywords" content="HTML,CSS,XML,JavaScript">
</head>
[...]
</html>
```

<title> = titolo

Ad ogni pagina HTML può essere associato un titolo che ne descrive il contenuto e che viene riportato nella barra del titolo della finestra del browser di Internet. Il titolo è usato

principalmente per identificare il documento (per esempio quando si cerca un documento tra tanti attraverso un motore di ricerca oppure quando una pagina viene inserita in un bookmark di siti preferiti) è opportuno quindi che ogni tiolo sia diverso dagli altri all'interno dell'insieme di pagine che formano un sito e che sia formato da una frase significativa per ricordarne il contenuto.

Il titolo viene racchiuso tra la coppia di tag **<title> </title>** e viene mostrato sulla barra del titolo della finestra oppure sulle linguette delle pagine Web se, nel browser, esse sono organizzate a schede.

<meta> tag

Esempio

```
<head>
   <meta name="description" content="Free Web tutorials">
   <meta name="keywords" content="HTML,CSS,XML,JavaScript">
   <meta name="author" content="Web Master">
   <meta charset="UTF-8">
</head>
```

Nella sezione di intestazione della pagina HTML, tra <head> e </head>, sono inseriti anche i **meta tag**: sono i tag che contengono i **metadati** della pagina, cioè informazioni sul contenuto della pagina, le parole chiave e l'autore. I meta tag sono identificati dal tag **<meta>**.

I valori più usati per l'attributo **name** sono:

- **description** per la descrizione;
- **keywords** per le parole chiave;
- **author** per l'autore

Un'importante informazione riguarda la specifica della codifica della pagina. Se non si specifica la codifica corretta può accadere che alcuni caratteri non standard, per esempio le lettere accentate, non vengano visualizzate correttamente.

Per specificare la codifica **UTF-8**, che contiene tutti i caratteri occidentali, si utilizza il codice:

```
<meta charset="UTF-8">
```

<body> = corpo del documento

Contiene tutti gli elementi della pagina che verranno effettivamente visualizzati a video: testo, immagini, applet Java, codice Javascript, e tutti quei contenuti multimediali che vengono mostrati sullo schermo. Il tag **<body>**, può essere utilizzato in forma semplice oppure se ne possono specificare alcuni attributi e i relativi valori.

ELENCO DEI TAG HTML (New Nuovi tag introdotti in HTML5)

Tag	Descrizione
<!--...-->	**Commento**
<!DOCTYPE>	**Tipo di documento**
<a>	**Collegamento ipertestuale**
<abbr>	Abbreviazione
<address>	Informazioni di contatto per l'autore/proprietario della pagina
<area>	Area all'interno di una mappa-immagine (image-map)
<article> New	Sezione indipendente con una connotazione semantica ben definita
<aside> New	Rappresenta una sezione che include un contenuto che è collegato a quanto trattato nella pagina, ma che è comunque distinto da esso
<audio> New	Definisce un contenuto audio
<b>	**Specifica lo stile del testo in neretto (o grassetto)**
<base>	**Specifica l'URL di base per tutti i percorsi relativi presenti nella pagina**
<bdi> New	Rappresenta una porzione di testo che deve essere isolato dal resto del contenuto per essere formattato in modo bidirezionale

<bdo>	Selezione di testo in cui la lettura avviene in direzione contraria rispetto al testo circostante(**bid**irectional **o**verride=sovraimpostazione bidirezionale)
<blockquote>	Indica una lunga citazione prelevata da una fonte esterna
<body>	**Definisce il corpo della pagina**
** **	**Definisce un'interruzione di riga (interlinea singola)**
<button> New	Crea un pulsante cliccabile
<canvas> New	Rappresenta uno spazio a due dimensioni per il rendering dinamico di immagini bitmap (ad esempio grafica) tramite script (in genere Javascript)
<caption>	**Definisce una didascalia per una tabella**
<cite>	Definisce il titolo di un lavoro (ad esempio un libro, un film, una canzone)
<code>	Definisce una sezione di testo per rappresentare un listato di codice
<col>	Specifica le proprietà di ogni colonna all'interno dell'elemento <colgroup>
<colgroup>	Definisce un gruppo logico di colonne in una tabella
<command> New	Comando che può essere invocato dall'utente, all'interno di un elemento menu o della pagina
<datalist> New	**Specifica una lista di opzioni predefinite per i campi di input**
<dd>	Indica la descrizione di un elemento in un elenco di definizioni
<del>	Indica una sezione di testo eliminato rispetto ad una versione precedente
<details> New	Rappresenta un insieme di informazioni o controlli aggiuntivi che possono essere mostrati a richiesta nella pagina
<dfn>	Rappresenta una definizione che viene descritta nel paragrafo che segue
<dialog> New	Crea un a dialog box or window
<div>	**Definisce una sezione (contenitore) in una pagina**
<dl>	Indica un elenco di definizioni (definition list)
<dt>	Indica un termine in un elenco di definizione (definition term)
<em>	Formatta una porzione di testo in modo enfatizzato (corsivo)
<embed> New	Incorpora un oggetto/applicazione esterna (non-HTML) nella pagina
<fieldset>	**Crea un riquadro che raggruppa gli elementi correlati in un form**
<figcaption>	Rappresenta la didascalia per un elemento figure ed è opzionale
<figure> New	Rappresenta un blocco distinto dal testo principale, pensato per contenere immagini, diagrammi, esempi, etc.
<footer> New	Rappresenta un blocco di chiusura all'interno di una sezione o pagina
<form>	**Definisce un modulo di inserimento dati da parte dell'utente**
<h1> to <h6>	**Definisce un titolo di varia grandezza (dal più grande al più piccolo)**
<head>	**Definisce l'area di intestazione della pagina html contenente informazioni sulla pagina (titolo, foglio di stile, meta tag, script, etc.)**
<header> New	Rappresenta un blocco di intestazione per una pagina o una sezione
<hgroup> New	Intestazione di una sezione e raggruppa uno o più elementi <h1> … <h6>
<hr>	Crea una linea che separa il contenuto all'interno di una pagina
<html>	**Definisce la radice di una pagina HTML**
<i>	Rappresenta una porzione di testo in un modo alternativo (stile italico)
<iframe>	Crea un frame inline nella pagina
<img>	**Definisce un'immagine**
<input>	**Crea un campo di input**
<ins>	Indica una sezione di testo che è stata inserita rispetto ad una versione precedente
<kbd>	Riproduce il testo con un carattere a spaziatura fissa
<keygen> New	Genera una coppia di chiavi numeriche all'interno di un form
<label>	**Definisce un'etichetta per un campo di input**
<legend>	**Definisce una didascalia per gli elementi <fieldset>, < figure>, <details>**
** **	**Definisce un elemento di una lista (ordinata e non ordinata)**
<link>	**Definisce il collegamento tra una pagina e una risorsa esterna (viene usato prevalentemente per collegare un file .css alla pagina HTML)**
<map>	Specifica una collezione di aree sensibili per una mappa immagine lato client

<mark> New	Mette in risalto una porzione di testo rispetto al contenuto (sfondo giallo)
<menu>	Rappresenta un menu contestuale, una toolbar o un elenco di comandi
<meta>	**Rappresenta una meta informazione relativa ad una pagina HTML**
<meter> New	Rappresenta una misura scalare in un intervallo noto, o un valore frazionario
<nav> New	Rappresenta una sezione che contiene una serie di link che permettono di accedere ad altre pagine o ad altre sezioni della pagina corrente
<noscript>	Definisce un contenuto alternativo per i browser che non supportano lo scripting
<object>	Definisce un oggetto incorporato nella pagina (audio, video, flash, etc.)
** **	**Crea una lista ordinata**
<optgroup>	**Definisce un gruppo di opzioni correlate all'interno di un elemento select**
<option>	**Definisce un'opzione all'interno di un elemento select (drop-down list)**
<output> New	Restituisce il risultato di un calcolo
<p>	**Definisce un paragrafo**
<param>	Definisce le proprietà dell'oggetto all'interno dell'elemento `object`
<pre>	Riproduce il testo con un carattere a spaziatura fissa
<progress> New	Rappresenta lo stato di avanzamento di un processo
<q>	Indica una breve citazione che viene racchiusa tra virgolette
<rp> New	All'interno del tag `ruby` , permette di isolare le parentesi poste attorno al testo associato a un ideogramma dal testo stesso (ruby parenthesis)
<rt> New	All'interno del tag `ruby`, denota la componente testuale associata a un ideogramma (ruby text)
<ruby> New	Rappresenta una porzione di testo espresso tramite ideogrammi, utilizzati principalmente nelle lingue orientali
<s>	Rappresenta il testo non più corretto utilizzando il carattere barrato
<samp>	Riproduce il testo come listato di codice in un esempio (font spaziatura fissa)
<script>	Definisce uno script lato client (client-side script)
<section> New	Sezione generica della pagina, senza una connotazione specifica
<select>	**Crea un menu di opzioni (drop-down list)**
<small>	Formatta il testo con un carattere più piccolo rispetto a quello corrente
<source> New	Definisce molteplici percorsi per gli elementi <video> e <audio>
<span>	**Definisce una sezione di una pagina (in genere una porzione di testo)**
<strong>	Definisce un testo importante, impostando lo stile grassetto
<style>	**Definisce le informazioni di stile di una pagina (incorporato o inline)**
<sub>	Riproduce il testo con un carattere più piccolo rispetto a quello corrente (pedice)
<summary> New	Rappresenta l'informazione riepilogativa di un elemento `details`
<sup>	Riproduce il testo con un carattere più piccolo rispetto a quello corrente (apice)
<table>	**Crea una tabella**
<tbody>	**Racchiude il contenuto del corpo di una tabella**
<td>	**Definisce una cella in una tabella**
<textarea>	**Crea un campo di input multilinea (text area)**
<tfoot>	**Racchiude il contenuto della riga finale (footer) di una tabella**
<th>	**Definisce una cella di intestazione in una tabella**
<thead>	**Racchiude il contenuto della riga iniziale (intestazione) di una tabella**
<time> New	Rappresenta un orario oppure una data opzionalmente con ora e fuso
<title>	**Titolo di una pagina che appare nella barra del titolo della finestra del browser**
<tr>	**Definisce una riga in una tabella**
<track> New	Riproduce il testo che accompagna gli elementi <video> e <audio>
<u>	Riproduce il testo con stile diverso rispetto al testo normale (sottolineato)
** **	**Crea una lista non ordinata**
<var>	Definisce una variabile
<video> New	Definisce un video
<wbr> New	Definisce un'interruzione di riga opzionale

N.B.: in neretto gli elementi trattati in questo manuale.

<!--...-->

Esempio

```
<!--This is a comment. Comments are not displayed in the browser-->
<p>This is a paragraph.</p>
```

Definizione ed uso

Una strategia importante per rendere il nostro codice più leggibile è quella di inserire dei **“commenti”** nei punti più significativi: si tratta di indicazioni significative per il webmaster, ma invisibili al browser. L’inserimento di commenti in punti specifici del documento ci permette di mantenere l’orientamento anche in file molto complessi e lunghi. La sintassi è la seguente:

```
<!-- questo è un commento -->
```

e ci permette di “commentare” i vari punti della pagina. Ad esempio:

```
<!-- menu di sinistra -->
<!-- barra in alto -->
<!-- eccetera -->
```

Contrassegnare la chiusura dei tag: i div

Tra le pratiche più di uso dei commenti, c’è quella di **contrassegnare l’inizio e la fine di un tag**: spesso si tratta di div che definiscono la struttura (layout) della pagina. Aggiungendo un commento che ne riporti l’id, ad esempio, sappiamo sempre in che area stiamo posizionando il nostro codice e rendiamo più leggibile il markup della pagina:

```
<div id="main"> <!-- inizio main -->
   <p>
      ...
      contenuto
   </p>
   <div id="sidebar"> <!-- inizio sidebar -->
      <ul>
         <li>... </li>
         <li>contenuto</li>
      </ul>
   </div> <!-- fine sidebar -->
</div> <!-- fine main -->
```

<!DOCTYPE>

Esempio

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

Come dice il W3C non esiste solo un tipo di HTML, ma ce ne sono molti: HTML 4.01 Strict, HTML 4.01 Transitional, XHTML 1.0 Strict, HTML5 e altri ancora. Tutti questi tipi sono definiti dalle loro rispettive specifiche, ma la loro struttura è anche definita da un linguaggio chiamato **DTD** usato per definire i componenti ammessi e la struttura dello stesso documento.

Visto il variare di questo tipo di specifiche è necessario usare il giusto **DOCTYPE** differenziandolo nei diversi tipi di documenti.

Cos'è il DTD?

Il **DTD** (***Document Type Definition***) è un file essenziale per la validazione della pagina HTML , è richiamato all'interno del DOCTYPE ed ha le seguenti caratteristiche:

1. Definisce gli **elementi** utilizzabili all'interno di un documento, decide quello che si può usare e quello che non si può: come un vocabolario per l'HTML.
2. Definisce determinate **regole** per ogni elemento: ne regola la quantità, decide se è obbligatorio o opzionale, determina l'ordine e se può avere elementi al suo interno.
3. Dichiara gli **attributi** definibili per ogni elemento e i valori che potrebbero assumere
4. Semplifica la dichiarazione di un documento e ne permette la **validazione**.

Cos'è e perché specificare il DOCTYPE?

Il **DOCTYPE** letteralmente "dichiarazione del tipo di documento" serve per dichiarare il DTD e quindi il tipo di (X)HTML utilizzato nella pagina.

Questo è essenziale per il corretto rendering di una normale pagina Html. Solo inserendo un DOCTYPE sarà possibile validare le pagine HTML con i validatori W3C.

Che tipo di DOCTYPE posso usare?

Per scegliere il DOCTYPE giusto, il consorzio W3C ha creato una tabella con tutti i DOCTYPE utilizzabili rispetto alla versione di HTML in uso dalla pagina web:

HTML 4.01 Strict	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 Transitional	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 Strict (quick reference)	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 Transitional	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1 – DTD	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
XHTML Basic 1.1 (quick reference)	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
HTML5	**<!DOCTYPE HTML>**

Per esempio se la pagina è un documento HTML5 il DOCTYPE sarà:

```
<!DOCTYPE html>
```

In altre parole, il **DOCTYPE** di una pagina web non è altro che la dichiarazione del tipo di documento corrente ed è stato concepito per comunicare al browser quale versione di HTML viene utilizzata; per questo motivo esso deve essere specificato all'inizio di un documento HTML. Questo tipo di informazione non è secondaria: senza di essa, per esempio, le regole di stile CSS non avrebbero effetto sulla formattazione.

Esempio

```
<a href="http://www.w3schools.com">Visit W3Schools.com!</a>
```

I link e l'ipertestualità

Una delle caratteristiche che ha fatto la fortuna del web è l'essere costituito non da **testi** ma da **ipertesti** (un'altra delle caratteristiche che hanno fatto grande il web è senz'altro la possibilità di interagire, ma questo è un altro discorso).

I link sono "il ponte" che consente di passare da un testo all'altro. In quanto tali, i link sono formati da due componenti:

il contenuto che "nasconde" il collegamento (non importa se si tratta di testo o di immagine)	È la parte visibile del link, e proprio per questo l'utente deve essere sempre in grado di capire quali sono i collegamenti da cliccare all'interno della pagina
la risorsa verso cui il collegamento punta	Si tratta di un'altra pagina (sullo stesso server o su un server diverso), oppure è un collegamento interno a un punto della pagina stessa

Di solito per spiegare che cosa sono i link si utilizza la metafora dell'ancora con "la testa" all'interno del documento stesso, e la "coda" in un altro documento (o all'interno di un altro punto del documento stesso).

Link che puntano ad altri documenti

Ecco la sintassi per creare un link con riferimento a un sito web:

```
Le risorse per webmaster sono su
<a href="http://www.html.it/">HTML.IT</a>
```

Come si può intuire la testa della nostra àncora è il testo "HTML.IT", mentre la coda, cioè la destinazione (specificata dall'attributo **`href`**) è il sito web verso cui il link punta, cioè http://www.html.it.

È indifferente che la destinazione dell'ancora sia una pagina HTML di un sito, un'immagine, un file pdf , un file zip, o un file exe: il meccanismo del link funziona allo stesso modo indipendentemente dal tipo di risorsa; poi il browser si comporterà in modo differente a seconda della risorsa. Ad esempio:

Immagine .gif, .jpg, .png	Viene visualizzata nel browser
Documento .html, .pdf, .doc	La pagina è visualizzata nel browser. Nel caso dei documenti **.doc** e **.pdf** l'utente deve avere installato sul proprio pc l'apposito plugin (nella maggior parte dei casi è sufficiente che abbia installato rispettivamente Microsoft Word e Adobe Acrobat Reader). Se non è installato il plugin il sistema chiederà all'utente se salvare il file.
File .zip, file .exe	Viene chiesto all'utente di scaricare il file **NOTA bene: per motivi di sicurezza non è possibile eseguire un file ".exe" direttamente dal web; l'utente dovrà sempre prima scaricarlo sul proprio PC.**

Potete anche specificare un indirizzo e-mail. In questo caso si aprirà direttamente il client di posta dell'utente con l'indirizzo e-mail pre-impostato. La sintassi è la seguente:

```
<a href="mailto:tuaMail@nomeTuoSito.it">Mandami una e-mail</a>
```

I percorsi assoluti e relativi

Percorsi assoluti

Fino a quando ci troviamo nella condizione di creare un sito web di dimensioni ridotte (poche pagine) non avremo problemi di complessità, e possiamo anche ipotizzare di lasciare tutti i nostri file in una medesima cartella. È evidente però che – man mano che il nostro sito web cresce – avremo bisogno di un maggior ordine.

Si presenterà allora l'esigenza di inserire le immagini del sito in una cartelle diverse (in modo da averle tutte nella medesima locazione), e magari sarà opportuno dividere il sito in varie sezioni, in modo da avere tutti i documenti dello stesso tipo all'interno di un contesto omogeneo.

I siti web sono dunque organizzati in strutture ordinate: non a caso si parla di **albero di un sito**, per indicare la visualizzazione della struttura alla base del sito.

Poiché l'organizzazione di un sito in directory e sottodirectory è una cosa normalissima, dobbiamo imparare a muoverci tra i vari file che costituiscono il sito stesso, in modo da essere in grado di creare collegamenti verso i documenti più reconditi, destreggiandoci tra le strutture più ramificate.

Per farlo esistono due tecniche:

- **indicare un percorso assoluto**
- **indicare un percorso relativo**

Nel caso in cui il documento a cui vogliamo puntare si trovi in una particolare directory del sito di destinazione, con i percorsi assoluti non abbiamo che da indicare il percorso per esteso.

Se esaminiamo:

```
Leggi le risorse sui
<a href="http://www.html.it/css/index.html">fogli di stile</a>
```

Possiamo vedere chiaramente che il link indica un percorso assoluto e fa riferimento ad una particolare directory. Nella fattispecie:

http://	Indica al browser di utilizzare il protocollo per navigare nel web (l'http)
www.html.it/	Indica di fare riferimento al sito www.html.it
css/	Indica che la risorsa indicata si trova all'interno della cartella "css"
index.html	Indica che il file da collegare è quello chiamato "index.html"

Insomma, per creare un collegamento assoluto è sufficiente fare riferimento all'url che normalmente vedete scritto nella barra degli indirizzi. I percorsi assoluti si usano per lo più, quando si ha la necessità di fare riferimento a risorse situate nei siti di terze persone.

Percorsi relativi

Spesso vi troverete tuttavia a fare riferimento a documenti situati nel vostro stesso sito, e – se state sviluppando il sito sul vostro computer di casa (cioè "in locale") – magari non avete ancora un indirizzo web, e non sapete di conseguenza come impostare i percorsi. È utile allora capire come funzionano i percorsi relativi.

I percorsi relativi fanno riferimento alla posizione degli altri file rispetto al documento in cui ci si trova in quel momento. Per linkare due pagine che si trovano all'interno della stessa directory è sufficiente scrivere:

```
<a href="paginaDaLinkare.html">collegamento alla pagina da linkare nella
stessa directory della pagina presente</a>
```

Poniamo ora di trovarci in una situazione di questo genere:

Figura 1. Riferimento a pagina di una sottodirectory

Dalla pagina “index.html” vogliamo cioè far riferimento al file “interna.html”, che si trova all’interno della directory “interna”, che a sua volta si trova all’interno della directory “prima”.

La sintassi è la seguente:

```
<a href="prima/interna/interna.html">Visita la pagina interna</a>
```

Vediamo adesso l’esempio opposto: dalla pagina interna vogliamo far riferimento a una pagina (“index.html”) che si trova più in alto di due livelli:

Figura 2. Riferimento a pagina in una directory di livello superiore

La sintassi è la seguente:

```
<a href=”../../index.html”>Visita la pagina interna</a>
```

Come si vede, con i percorsi relativi valgono le seguenti regole generali:

Per far riferimento a un file che si trovi all’interno della stessa directory basta linkare il nome del file	collegamento alla pagina
Per far riferimento a un file contenuto in una cartella di livello inferiore alla posizione corrente, basta nominare la cartella seguita dallo "slash", e poi il nome del file. Secondo la formula: **cartella/nomeFile.html**	Visita la pagina interna
Per tornare su di un livello, è sufficiente utilizzare la notazione: **../nomeFile.html**	Visita la pagina interna

Grazie a questi accorgimenti potete agevolmente navigare all’interno delle directory del vostro sito: se ce ne fosse bisogno potrete per esempio tornare su di un livello rispetto alla posizione del file, scegliere un’altra cartella, e poi scegliere un altro file:

```
../altraCartella/nuovoFile.html
```

Approfondimenti

A volte potrete incontrare la notazione:

```
Leggi le risorse sui <a href="/css/index.html">fogli di stile</a>
```

Se il vostro sito è all'interno di un server Unix (ma la sintassi funziona anche in sistemi Windows, basta che non siano in locale), questa notazione non deve stupirvi: il carattere '/' indica la directory principale del sito, altrimenti detta "root". Dunque <a href="/css/index.html"> è un altro modo di esprimere i percorsi assoluti all'interno del proprio sito.

Un'altra cosa importante da sapere è che quando metterete il vostro sito all'interno dello spazio web, l'indicazione della index all'interno di una directory è facoltativa. Al posto di questo:

```
http://www.html.it/css/index.html
```

è sufficiente indicare la directory:

```
http://www.html.it/css/
```

Verificate solo con il vostro gestore dello spazio web (cioè "hosting"), se le pagine index della directory devono avere forma index.html, index.htm, index.asp, index.php, home.asp, o altro.

Consigli per i nomi dei file

Quando mettere nel web il vostro sito internet, vi accorgerete che esistono due famiglie di sistemi operativi: Windows e Unix. Questi due sistemi operativi utilizzano differenti modi per gestire i file, dunque alcuni accorgimenti sono necessari:

- è consigliabile non lasciare spazi vuoti nei nomi dei file (gli spazi vuoti non sempre vengono interpretati correttamente), meglio ovviare a questa necessità con un "trattino basso" (cioè "_"). Ad esempio: mio_file.html
- maiuscole e minuscole possono fare la differenza (in ambiente Unix spesso la fanno), quindi controllate il modo in cui avete scritto i file

Inoltre quando create un collegamento state attenti a non avere una notazione simile a questa:

```
<a href="file:///C|percorsonomeFile.html">testo</A>
```

significa che state facendo un riferimento (assoluto) al vostro stesso computer: chiaro che quando metterete i file nel vostro spazio web, le cose non funzioneranno più.

I link interni o ancore

È possibile anche creare un indice interno al documento, utilizzando le àncore. Ciascuna àncora può avere infatti un id:

```
<a id="primo">Stiamo per esaminare la struttura…. Eccetera…</a>
```

Da notare che in mancanza dell'attributo che indica il collegamento (href) le àncore non vengono viste come link, ma la loro formattazione è indistinguibile dal "normale" testo.

In un ipotetico indice è allora possibile far riferimento all'àncora presente all'interno del documento attraverso un link che punti ad essa:

```
<a href="#primo">vai al primo paragrafo</a>
```

il cancelletto indica che il collegamento deve cercare un àncora chiamata "primo" all'interno della pagina stessa.

Se non si specifica il nome dell'àncora a cui si vuol puntare, viene comunque creato un link che punta ad inizio pagina (viene cercata un'àncora il cui nome non è specificato). Questo infatti è un ottimo escamotage per creare link "vuoti" (in alcuni casi vi occorreranno). Ad esempio:

```
<a href="#">link vuoto</a>
```

Per creare un indice interno alla pagina si procede dunque in due fasi distinte:

- creazione dell'ancora a cui puntare (<a id="mioNome">)
- creazione del collegamento all'ancora appena creata e riferimento
- attraverso il cancelletto (<a href="#mioNome">)

È bene non confondere le due fasi.

Gli attributi dei link

target

È anche possibile specificare in quale finestra la pagina linkata deve essere aperta: di default infatti la pagina viene aperta all'interno del documento stesso:

```
<a target="_self" href="http://www.html.it">visita HTML.IT</a>
```

ma è possibile specificare che la pagina sia aperta in una nuova finestra:

```
<a target="_blank" href="http://www.html.it">visita HTML.IT</a>
```

title

L'attributo **title** è molto importante, e serve per specificare un testo esplicativo per l'elemento a cui l'attributo è riferito (il title si può infatti utilizzare anche per elementi differenti dalle ancore). Questa spiegazione addizionale favorisce l'accessibilità del sito anche ai disabili, alle persone per esempio che hanno disturbi alla vista. Se lasciate il cursore del mouse per qualche secondo su un collegamento dotato di title, vedrete comparire una specie di etichetta con il testo specificato nel `title`:

```
<a title="in HTML.it puoi trovare risorse per webmaster"
    href="http://www.html.it/" target="_blank" >
    Visita HTML.it
</a>
```

L'attributo `title` è anche utilissimo per migliorare la propria presenza nei motori di ricerca, che ne vanno a leggere il contenuto.

hreflang

Con "hreflang" si indica la lingua del documento: si tratta di un attributo che migliora l'accessibilità del sito, oltre ad essere potenzialmente utile per i motori di ricerca (l'attributo può essere utilizzato ad esempio per specificare la presenza di una sezione del proprio sito in lingua inglese):

```
Nel sito del <a href="http://www.w3c.org/" hreflang="eng"
target="_blank">World Wide Web Consortium</a> puoi trovare le specifiche
dell'HTML in lingua inglese
```

accesskey

Le **accesskey** sono delle scorciatoie "da tastiera" che potete utilizzare nel vostro sito. Si tratta di scegliere delle lettere della tastiera che – quando vengano digitate dall'utente – permettono di andare direttamente a determinate pagine. Per esempio potreste specificare che:

```
<a href="http://www.html.it/" accesskey="h" target="_blank" >Torna
all'home page di HTML.it</a>
```

In questa pagina digitando "**ALT + h + invio**" con Internet Explorer, oppure direttamente "**h + invio**" con Mozilla si accede direttamente all'home page di HTML.it. Si tratta di un'altra tecnica per migliorare l'accessibilità, ma un uso improprio e indiscriminato di questa tecnica può risultare davvero deleterio per la navigazione. Diciamo che le accesskey dovrebbero essere riservate per la navigazione dei menu che portano alle parti principali del sito.

<base>

Esempio

```
<head>
<base href="http://www.w3schools.com/images/" target="_blank">
</head>
<body>
<img src="stickman.gif" width="24" height="39" alt="Stickman">
<a href="http://www.w3schools.com">W3Schools</a>
</body>
```

I percorsi relativi fanno di norma riferimento alla directory in cui si trova il file HTML che stiamo scrivendo. Se tuttavia vogliamo far riferimento a un differente percorso **per tutti i percorsi relativi**, possiamo farlo specificandolo come attributo **href** del tag **<base>**, che va incluso nella sezione **head** del documento. Ad esempio con:

```
<base href="http://www.mioSitoWeb.com/miaCartella">
```

specifico che d'ora in poi tutti i percorsi relativi faranno riferimento al percorso indicato. E poi nel documento potrò scrivere:

```
<a href="mioFile.html">collegamento al mio file</a>
```

sicuro che farà riferimento a:

```
http://www.mioSitoWeb.com/miaCartella/mioFile.html
```

Si tratta di una caratteristica particolarmente utile quando bisogna mandare ad esempio delle mailing list in formato HTML: possiamo infatti utilizzare i percorsi relativi per sviluppare la pagina della mailing list in locale, e mantenerli inalterati grazie all'utilizzo di questo tag. Grazie ad esso siamo infatti sicuri che anche l'utente che riceverà la mail potrà visualizzare le immagini e i link con un percorso corretto.

Gli attributi del tag <base>

href, target

Oltre all'attributo **href**, con l'attributo **target** specifichiamo in quale finestra saranno aperti tutti i collegamenti ipertestuali presenti nella pagina corrente. Per i valori di questo attributo si rimanda a quanto già detto per l'elemento **<a>**.

<h1>...<h6>

Esempio

```
<h1>titolo 1 </h1>
<h2>titolo 2 </h2>
<h3>titolo 3 </h3>
<h4>titolo 4 </h4>
<h5>titolo 5 </h5>
<h6>titolo 6 </h6>
```

In questo tag la lettera iniziale "h" sta per "heading", cioè titolo. Le grandezze previste sono sei: da **<h1>**, il più importante, si va via via degradando fino al più piccolo **<h6>**. Il tag <hx> (sia esso h1 o h6) risulta formattato in grassetto e lascia una riga vuota prima e dopo di sé.

<div>

Esempio

```
<div>
   <h3>This is a heading</h3>
   <p>This is a paragraph.</p>
</div>
```

Il tag (elemento) <div> è usato per definire una sezione del documento. Non ha nessuna funzione di formattazione sul testo. è invece una specie di contenitore per gli altri elementi (titoli, paragrafi, immagini):

```
<div>
   <h1>Titolo</h1>
   <p>Paragrafo</p>
</div>
```

<span>

Esempio

```
<p>My mother has <span style="color:blue">blue</span> eyes.</p>
```

Lo **<span>** è un contenitore generico che può essere annidato (ad esempio) all'interno dei <div>. Si tratta di un elemento **inline**, che cioè non va a capo e continua sulla stessa linea del tag che lo include.

Esempio

```
<div>
   <span>contenitore 1</span><span>contenitore 2</span>
</div>
```

Lo **<span>** è un elemento molto utilizzato soprattutto insieme ai fogli di stile (ad esempio per definire delle aree di testo particolari) e supporta tutti gli **attributi globali** HTML.

<p>

Esempio

```
<p>This is some text in a paragraph.</p>
```

Il paragrafo è l'unità di base entro cui suddividere un testo. Il tag **<p>** lascia una riga vuota prima della sua apertura e dopo la sua chiusura. Questo elemento supporta tutti gli **attributi globali** HTML.

<p>, <div>, <span>: usare gli elementi insieme

Le caratteristiche più evidenti di **<p>**, **<div>** e **<span>** sono quindi:

- **<p>** lascia spazio prima e dopo la propria chiusura;
- **<div>** non lascia spazio prima e dopo la propria chiusura, ma – essendo un elemento **di blocco** – va a capo;
- **<span>** – essendo un elemento **inline** – non va a capo.

Esempio

```
This text contains<br>a line break.
```

Il tag
 è un tag vuoto e viene usato per creare una semplice interruzione di riga.

Esempio

```
<img src="smiley.gif" alt="Smiley face" height="42" width="42">
```

Inserire le immagini

Naturalmente, all'interno delle nostre pagine Web possiamo inserire, oltre al testo, anche delle immagini: diagrammi e grafici, fotografie, e in genere immagini create con un programma di elaborazione grafica (come ad esempio Photoshop o Paint Shop Pro).

I formati ammessi nel Web sono sostanzialmente tre:

- **GIF** (**Graphic Interchange Format**): le GIF sono immagini con non più di 256 colori (dunque con colori piatti e senza sfumature), come grafici o icone;
- **JPG**: è l'acronimo del gruppo di ricerca che ha ideato questo formato (il **Joint Photographic Experts Group**), idoneo per le immagini di qualità fotografica;
- **PNG** (**Portable Network Graphic**). Il PNG è un tipo di immagine introdotto più recentemente, elaborato dal W3C (http://www.w3c.org) per risolvere i problemi di copyright del formato GIF (che è appunto proprietario); tuttavia oggi il PNG è letto oramai da tutti i browser e offre alcune caratteristiche che gli altri formati non hanno (come il supporto al canale alfa, caratteristica questa non ancora perfettamente supportata da ogni browser).

Non provate dunque a inserire un file ".psd" (è il formato nativo di Photoshop) all'interno della vostra pagina HTML: con grande probabilità il browser non vi caricherà il file che vorreste includere (dovete infatti prima convertire il file in uno dei formati sopra-indicati).

Inoltre è importante ricordare che il codice HTML fornisce delle indicazioni al browser su come visualizzare il testo e le immagini - ed eventualmente i video e i suoni - all'interno della pagina: il testo (come abbiamo visto) è scritto direttamente nel file HTML, le immagini invece sono caricate insieme alla pagina.

Attenzione dunque a non inserire immagini troppo pesanti (ricordatevi di ottimizzare sempre i file); bisogna evitare inoltre di sovraccaricare la pagina con troppe immagini, rallentandone il caricamento.

Per ottenere un sito web dalla grafica accattivante, spesso è sufficiente giocare con i colori dello sfondo e delle scritte.

Gli attributi delle immagini

src

L'attributo **`src`** indica il nome del file immagine eventualmente preceduto dal percorso sorgente:

```
<img src="logo.gif>
```

Resta valido il discorso sui percorsi relativi ed assoluti visto in precedenza. Avremo ad esempio:

Percorso relativo

```
<img src="images/logo.gif">
<img src="../images/logo.gif">
```

Percorso assoluto

```
<img src="http://www.html.it/img/logo.gif">
```

alt

L'attributo **`alt`** (obbligatorio) è utile per specificare il **testo alternativo** (**alternative text**), fintanto che l'immagine non viene caricata o nel caso in cui non lo sia affatto:

```
<img src="logo.gif" alt="Logo azienda">
```

Il testo alternativo dovrebbe:

- descrivere l'immagine, se questa contiene delle informazioni;
- descrivere la pagina di destinazione, se l'immagine incorpora un collegamento ipertestuale

Se l'immagine è puramente decorativa, l'attributo `alt` sarà impostato in questo modo:

```
<img src="smiley.gif" alt="">
```

L'attributo **alt** è di estrema utilità per rendere il **sito accessibile** a tutti gli utenti: i disabili che non sono in grado di vedere nitidamente le immagini sullo schermo potrebbero avere delle difficoltà, nel caso in cui l'attributo alt non sia specificato.

Gli ipo-vedenti e i non-vedenti sono infatti in grado di comprendere il contenuto delle immagini grazie a dei software appositi (gli **screen reader**) che "leggono" lo schermo tramite un programma di sintesi vocale. Non specificare il testo alternativo significa rendere impossibile la navigazione.

height, width

Gli attributi **height** e **width** indicano le dimensioni dell'immagine (altezza e larghezza) espresse in pixel.

```
<img src="logo.gif" alt="Logo azienda" height="69" width="224">
```

Dal momento che il browser normalmente non sa quali siano le dimensioni dell'immagine, finché questa non sia caricata completamente, è un'ottima abitudine quella di indicare già nel codice la larghezza (width) e l'altezza (height) dell'immagine: in questo modo si evita di vedere la pagina costruirsi man mano che viene caricata, poiché stiamo dando al browser un'idea dell'ingombro.

Per ridurre le dimensioni di visualizzazione dell'immagine è bene non agire sugli attributi `height` e `width` modificandone i rispettivi valori, perché l'immagine continuerebbe ad essere caricata nella sua versione originaria (di larghe dimensioni). In questi casi è opportuno rielaborare l'immagine con un apposito software (ad esempio PhotoShop) riducendone le dimensioni e salvandola in un nuovo file.

, <ul>,

**Esempio **

```
<ol>
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ol>
```

**Esempio **

```
<ul>
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ul>
```

Gli elenchi in HTML

Se abbiamo la necessità di inserire un elenco di termini, possiamo utilizzare le "liste", che sono sostanzialmente di due tipi:

- **Elenchi ordinati**
- **Elenchi non ordinati**

Entrambi i tipi di elenchi funzionano nel medesimo modo: si apre il tag, si elencano i vari elementi della lista (ciascuno con il proprio tag), si chiude il tag dell'elenco. La sintassi ha questa forma:

```
<elenco>
   <elemento>nome del primo elemento</elemento>
   <elemento>nome del secondo elemento</elemento>
</elenco>
```

A partire dalla versione HTML 5 il tag che individua l'elemento della lista non ha bisogno di chiusura (la sua chiusura, in questo caso, è opzionale).

Gli elenchi ordinati

Gli elenchi ordinati sono contraddistinti dall'enumerazione degli elementi che compongono la lista. Avremo quindi una serie progressiva ordinata e individuata da lettere o numeri.

Il tag da utilizzare per aprire un elenco ordinato è **`<ol>`** ("ordered list") e gli elementi sono individuati dal tag **`<li>`** ("list item"):

Codice	Output
Testo che precede la lista - primo elemento - secondo elemento - terzo elemento Testo che segue la lista	Testo che precede la lista 1. Primo elemento 2. Secondo elemento 3. Terzo elemento Testo che segue la lista

Il tag che individua l'elenco lascia una riga di spazio prima e dopo il testo che eventualmente lo circonda (come avviene per il **`<p>`**); fa eccezione però l'inclusione di un nuovo elenco all'interno di un elenco preesistente: in questo caso non viene lasciato spazio, né prima, né dopo. Gli elementi dell'elenco sono sempre rientrati di uno spazio verso destra: tutto questo serve a individuare in modo inequivocabile l'elenco.

Gli attributi degli elenchi ordinati

type

Lo stile di enumerazione visualizzata di default dal browser è quello numerico, ma è possibile indicare uno stile differente specificandolo per mezzo dell'**attributo type**. Ad esempio:

```
<ol type="a">
   <li>primo elemento</li>
   <li>secondo elemento</li>
   <li>terzo elemento</li>
</ol>
```

Gli stili consentiti sono:

Valore dell'attributo type	Descrizione	Codice	Output
type="1" (default)	**numeri arabi**	<ol type=”1”> - primo - secondo - terzo 	1. primo 2. secondo 3. terzo
type="a"	**alfabeto minuscolo**	<ol type=”a”> - primo - secondo - terzo 	a. primo b. secondo c. terzo

`type="A"`	**alfabeto maiuscolo**	`<ol type=”A”>`
		`<li>primo</li>`
		`<li>secondo</li>`
		`<li>terzo</li>`
		`</ol>`
			A. primo
			B. secondo
			C. terzo

`type="i"`	**numeri romani minuscoli**	`<ol type=”i”>`
		`<li>primo</li>`
		`<li>secondo</li>`
		`<li>terzo</li>`
		`</ol>`
			i. primo
			ii. secondo
			iii. terzo

`type="I"`	**numeri romani maiuscoli**	`<ol type=”I”>`
		`<li>primo</li>`
		`<li>secondo</li>`
		`<li>terzo</li>`
		`</ol>`
			I. primo
			II. secondo
			III. terzo

start

L’attributo `start` indica il valore di partenza del primo elemento dell’elenco ordinato.

```
<ol start="10">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ol>
```

reversed New

L’attributo `reversed` (introdotto in HTML 5) indica l’ordinamento degli elementi in senso decrescente.

```
<ol reversed>
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
</ol>
```

Gli elenchi non ordinati

Gli elenchi non ordinati sono individuati dal tag **<ul>** (“unordered list”), e gli elementi dell’elenco sono contraddistinti anch’essi dal tag **<li>** (in buona sostanza si tratta di quello che i programmi di videoscrittura chiamano elenchi puntati):

```
<ul>
  <li>primo elemento</li>
  <li>secondo elemento</li>
  <li>terzo elemento</li>
</ul>
```

Da notare inoltre che il tipo di segno grafico, varia in automatico al variare dell’annidamento della lista. Ad esempio:

Codice

```
<ul>
  <li>primo della 1a lista</li>
  <li>secondo della 1a lista
    <ul>
      <li>primo della 2a lista</li>
      <li>secondo della 2a lista
        <ul>
          <li>primo della 3a lista</li>
        </ul>
      </li>
      <li>terzo della 2a lista</li>
    </ul>
  </li>
</ul>
```

Output

- primo della 1^a lista
- secondo della 1^a lista
  - primo della 2^a lista
  - secondo della 2^a lista
    - primo della 3^a lista
  - terzo della 2^a lista

<table>, <tr>, <th>, <td>, <caption>

Esempio

```
<table border="1">
   <caption>Elenco delle spese mensili</caption>
   <thead>
      <tr>
         <th>Mese</th>
         <th>Spesa</th>
      </tr>
   </thead>
   <tfoot>
      <tr>
         <td>Totale</td>
         <td>€ 180</td>
      </tr>
   </tfoot>
   <tbody>
      <tr>
         <td>Gennaio</td>
         <td>€ 100</td>
      </tr>
      <tr>
         <td>Febbraio</td>
         <td>€  80</td>
      </tr>
   </tbody>
</table>
```

Tabella: struttura di base

I tag necessari per creare una tabella sono:

Tag	Descrizione
<table>	crea la tabella
<tbody>	"table-body": apre il corpo della tabella
<thead>	"table-header": apre il corpo della riga di intestazione della tabella
<tfoot>	"table-footer": apre il corpo della riga finale della tabella
<tr>	"table-row": indica l'apertura di una riga

<th>	“table-head”: indica una cella all’interno della prima riga di intestazione
<td>	“table-data”: indica una cella all’interno di una riga successiva alla prima
<caption>	Didascalia della tabella posizionata al di sopra della stessa allineata al centro

In questi nostri primi esempi presupponiamo che il numero delle celle all’interno di ciascuna riga sia costante: ogni riga avrà cioè lo stesso numero di celle. Ci sono dei metodi per variare il numero delle celle all’interno di una riga, ma li vedremo in seguito.

Gli attributi della tabella (<table>)

border

L’attributo **border** permette di specificare lo spessore in pixel del bordo delle tabelle. Esempio:

```
<table border="2">
```

Lo useremo in questi esempi, altrimenti non percepiremmo la struttura di quanto stiamo costruendo. Ecco un primo esempio di tabella:

```
<table border="1">
   <tr>
      <th>prima cella</th>
      <th>seconda cella</th>
   </tr>
   <tr>
      <td>terza cella</td>
      <td>quarta cella</td>
   </tr>
</table>
```

che viene visualizzato così:

prima cella	seconda cella
terza cella	quarta cella

Gli attributi della cella (<th>, <td>)

colspan, rowspan

È possibile raggruppare le delle all'interno delle colonne in modo da avere ad esempio una riga da 2 colonne e un’altra da 3. Per ottenere questo risultato è necessario specificare che una cella deve occupare il posto di 2 (o più colonne). In questo caso si utilizza l’attributo **colspan** sul <td>, specificando come valore il numero di celle che devono essere occupate. Ad esempio:

<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td colspan="2">celle unite</td></tr></table>

Il cui codice corrispondente è:

```
<table border="1">
   <tr>
      <th>1</th>
      <th>2</th>
      <th>3</th>
   </tr>
   <tr>
      <td>4</td>
      <td colspan="2">
         <b>celle unite</b>
      </td>
   </tr>
</table>
```

Tramite l’attributo **rowspan** (da riferirsi sempre a <td>) è invece possibile creare delle celle che occupino più di una riga. Ad esempio:

<table>
<tr><td>1</td><td rowspan="2">celle unite</td><td>3</td></tr>
<tr><td>4</td><td>6</td></tr>
</table>

il cui codice corrispondente è:

```
<table border="1">
   <tr>
      <td>1</td>
      <td rowspan="2">
         <b>celle unite</b>
      </td>
      <td>3</td>
   </tr>
   <tr>
      <td>4</td>
      <td>6</td>
   </tr>
</table>
```

Annidare le tabelle

È anche possibile annidare le tabelle le une dentro le altre. Come in questo esempio:

```
<table border="1">
   <tr>
      <td>A</td>
      <td>B</td>
   </tr>
   <tr>
      <td>C</td>
      <td>
         <table border="1">
            <tr>
               <td>D1</td>
               <td>D2</td>
               <td>D3</td>
            </tr>
            <tr>
               <td>D4</td>
               <td>D5</td>
               <td>D6</td>
            </tr>
         </table>
      </td>
   </tr>
</table>
```

che dà come risultato:

<table>
<tr><td>A</td><td>B</td></tr>
<tr><td>C</td><td><table><tr><td>D1</td><td>D2</td><td>D3</td></tr><tr><td>D4</td><td>D5</td><td>D6</td></tr></table></td></tr>
</table>

<form>

Esempio

```
<form action="demo_form.asp" method="get">
   Nome: <input type="text" name="nome"><br>
   Cognome: <input type="text" name="cognome"><br>
   <input type="submit" value="Invia">
</form>
```

I siti Web non sono soltanto un insieme di pagine da leggere con testo e immagini, ma rappresentano uno strumento per interagire con gli utenti per esempio un’azienda può utilizzare un sito Web non solo per presentare i suoi prodotti o servizi, ma anche per raccogliere ordini o richieste di informazioni sui prodotti oppure per effettuare vendite e transazioni commerciali.

Per l’invio di informazioni da parte dell’utente si usano i form (moduli) che in HTML sono racchiusi tra la coppia di tag **<form>** … **</form>**.

Gli attributi del form

action

L’attributo **action** indica l’URL del programma o della pagina di risposta che processerà i dati.

```
<form id=”datiUtenti” action=”paginaRisposta.aspx”>
```

method

Quando creiamo un form possiamo scegliere due metodi di invio: **GET** e **POST**.

Con il metodo **GET** la pagina di risposta viene contattata e i dati vengono inviati in un unico step. Nell’URL della pagina di risposta potremo allora vedere tutti i parametri nella barra degli indirizzi (più precisamente nella “**query string**“, cioè nella “**stringa di interrogazione**“) secondo questa forma:

```
paginaRisposta.php?nome=Paolo&cognome=Rossi&datiInviati=prova+invio
```

i dati (nella forma **nome del campo** = **valore del campo**) sono appesi alla pagina dopo il punto interrogativo.

Alcuni server, tuttavia, hanno delle limitazioni per quel che riguarda il metodo **GET** e non consentono di inviare form con valori superiori a **255 caratteri** complessivi. Il metodo **GET** è dunque particolarmente indicato per form con pochi campi e pochi dati da inviare. La sintassi perl’invio in get è:

```
<form id=”datiUtenti” action=”paginaRisposta.aspx” method=”get”>
```

Nel metodo **POST** invece l’invio dei dati avviene in due stepdistinti: prima viene contattata la pagina sul server che deve processare i dati, e poi vengono inviati i dati stessi. Per questo motivo i parametri non compaiono nella query string (dunque se non si desidera che i parametri siano mostrati all’utente questo metodo è preferibile).

In questo caso non ci sono limiti sulla lunghezza dei caratteri. La sintassi è:

```
<form id=”datiUtenti” action=”paginaRisposta.aspx” method=”post”>
```

name, id

Gli attributi **name** e **id** indicano il nome del form e identificano l’oggetto form nel foglio di stile o in una procedura Javascript.

```
<form name=”datiUtenti” id=”datiUtenti” action=”paginaRisposta.aspx”>
```

target

Grazie all'attributo **"target"** è possibile far aprire i dati del form nella pagina corrente:

```
<form name="datiUtenti" action="paginaRisposta.aspx" method="get"
target="_self">
```

oppure in una pagina differente:

```
<form name="datiUtenti" action="paginaRisposta.aspx" method="get"
target="_blank">
```

autocomplete New

Questo attributo previene un comportamento dei browser non sempre voluto: spesso i browser riempiono i campi da inserire in maniera automatica.

Questo comportamento è nella maggior parte dei casi un comportamento comodo, però in alcuni casi è fastidioso. Si pensi per esempio ai campi password o ai campi del codice della banca: probabilmente non vogliamo che il browser li completi in automatico.

Ecco che arriva in nostro soccorso l'attributo **`autocomplete`** che è un attributo enumerato. In particolare i valori che accetta sono:

- **on**: indica che il valore non è particolarmente sensibile e che il browser può compilarlo in maniera automatica;
- **`off`**: indica che il valore è particolarmente sensibile o con un tempo di scadenza (il codice di attivazione di un servizio, per esempio) e che quindi l'utente deve inserirlo manualmente ogni volta che lo compila;
- **nessun valore**: indica in questo caso di usare il valore di default scelto dal browser (normalmente on).

```
<form action="demo_form.aspx" method="get" autocomplete="on">
   Nome:<input type="text" name="nome"><br />
   E-mail: <input type="email" name="email"><br />
   <input type="submit">
</form>
```

novalidate New

Questo attributo si applica al tag **`form`** e permette di saltare tutte le validazioni dei tag che da esso discendono. **`novalidate`** è un attributo booleano.

Ecco un esempio di utilizzo:

```
<form novalidate>
   <label>Email:
      <input type="email" name="myEmail" required>
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

In questo caso non verrà controllato che il campo email sia in un formato valido, né che sia presente.

<input>

Esempio

```
<form action="demo_form.aspx">
   Nome: <input type="text" name="nome"><br>
   Cognome: <input type="text" name="cognome"><br>
   <input type="submit" value="Invia">
</form>
```

All'interno del form si possono inserire oggetti per acquisire i dati o le scelte dell'utente. I componenti del form per inserire un valore o effettuare una scelta vengono aggiunti utilizzando il tag **<input>**. Per specificare il tipo di input si utilizza l'attributo **type**.

Gli attributi di input

autocomplete New

Si veda quanto detto per il tag form.

autofocus New

L'attributo **autofocus** è un **attributo booleano** e serve a **impostare il focus su uno specifico elemento** del form appena la pagina è caricata. Un esempio canonico è quello della home page di Google: appena viene caricata il focus è automaticamente impostato sul campo per la ricerca.

Ovviamente solo un elemento per pagina può avere l'attributo autofocus.

Questo attributo deve essere usato con parsimonia in quanto alcuni utenti, come quelli che usano la barra spaziatrice per scorrere la pagina, potrebbero trovare questa costrizione fastidiosa, preferendo un atteggiamento più soft.

È per questo motivo che autofocus dovrebbe essere usato solo nelle pagine che contengono solamente (o principalmente) form (come per esempio pagine di login o di ricerca). Esempio:

```
<form action="/" method="get">
   <input type="text" name="nome" id="nome" autofocus>
   <input type="submit" value="Invia">
</form>
```

checked

È un attributo booleano e viene usato per impostare l'opzione predefinita per i tipi di input "checkbox" e "radio".

```
<form action="">
   <input name="html" type="checkbox" value="html" checked />
</form>
```

disabled

Questo attributo serve a disabilitare un campo di input, rendendolo inutilizzabile. Esempio:

```
<form action="demo_form.aspx">
   Nome: <input type="text" name="nome"><br>
   Cognome: <input type="text" name="cognome" disabled><br>
   <input type="submit" value="Invia">
</form>
```

form New

Un nuovo attributo che si può inserire in tutti gli `input` è appunto `form`, anche se sfortunatamente non è molto supportato e non esiste una vera e propria alternativa.

Questo nuovo attributo serve per specificare a quale `form`, o a quali `form`, l'input fa riferimento. Richiede come valore l'`id` del `form` a cui vogliamo che faccia riferimento, o nel caso di più `form`, gli `id` separati da uno spazio, " ".

```
<form action="/" method="get" id="myFormId">
   <input type="text" name="myname">
   <input type="submit" value="Invia">
</form>
<input type="text" name="mysurname" form="myFormId">
```

Nonostante l'input con `id mysurname` sia fuori dal form, inviando il form inviamo anche il suo valore grazie al suo attributo `form`.

L'uso di questo attributo è sconsigliato a meno di non conoscere con certezza il browser dell'utente che utilizzerà la nostra applicazione.

list New

```
<input list="browsers">

<datalist id="browsers">
   <option value="Internet Explorer">
   <option value="Firefox">
   <option value="Google Chrome">
   <option value="Opera">
   <option value="Safari">
</datalist>
```

L'attributo `list`, insieme con l'elemento HTML5 **`<datalist>`**, è utilizzato per fornire una funzione di "completamento automatico" ad un elemento del form. Permette, in altre parole, di fornire all'utente un elenco di opzioni predefinite da cui scegliere, offrendo al contempo la possibilità di inserire un valore non presente inizialmente nella lista. Mette insieme i vantaggi di una `select` e di un `input` di testo.

1. Visivamente abbiamo solo il nostro input e la `datalist` non è renderizzata in nessun modo (figura 1).
2. Se clicchiamo sull'input a cui la `datalist` è collegata, vengono presentate le opzioni della `datalist` (figura 2).
3. Se clicchiamo su un'opzione, l'input a cui la `datalist` è collegato assume il valore dell'opzione (figura 3).

Figura 1 – Datalist: stato iniziale

Figura 2 – Datalist: opzioni della datalist

Figura 3 – Datalist su Opera: valorizzazione dell'opzione

Per utilizzare **datalist** prima di tutto dobbiamo scrivere un input a cui collegare la nostra datalist. Per collegare l'input alla datalist basta impostare l'attributo list dell'input con l'id della datalist.

Per ogni suggerimento che vogliamo dare all'utente facciamo discendere da datalist un tag option, mettendo il suggerimento nell'attributo value. Il value dell'option non deve essere vuoto e l'option non deve essere settata su disabled per essere visualizzata.

Vediamo il codice:

```
<form name="gustiGelato" method="post" action="">
   <label>Gusti gelato:
      <input type="text" name="gusti" list="gusti">
      <datalist id="gusti">
         <option value="Cioccolato">
         <option value="Fragola">
         <option value="Pistacchio">
         <option value="Stracciatella">
         <option value="Amarena">
      </datalist>
   </label>
   <input type="reset" value="Cancella">
   <input type="submit" value="Invia">
</form>
```

maxlength, size

L'attributo maxlength indica il numero massimo di caratteri che l'utente può inserire, con size si esprimono invece le dimensioni del campo di testo (la larghezza è data dal numero di caratteri).

```
<input name="mioTesto" type="text" size="30" maxlength="50" />
```

L'attributo size funziona con i seguenti tipi di input: text, search, tel, url, email, password.

max New, min New, step New

HTML5 mette a disposizione un set di attributi specifici che servono a specificare delle limitazioni per il valore dei seguenti tipi di input: number, range, date, datetime, datetime-local, month, time e week. Questi attributi sono **min**, **max** e **step**.

- **min**: specifica il minimo valore permesso. Esempio: min="1", che permette solo di passare numeri da 1 in su.
- **max**: specifica il massimo valore permesso. Esempio: max="10", che permette solo di inviare numeri inferiori o uguali a 10. Il valore di questo attributo deve essere maggiore del valore dell'attributo min (se specificato).
- **step**: indica la granulosità che deve avere il value limitando i possibili valori passati. Il valore di step se specificato deve essere un numero (anche non intero) maggiore di zero oppure la stringa "any" (che equivale a non inserire l'attributo). Esempio: step=3, che influenza i valori inseriti passando valori come -3, 0, 3, 6.

```
<form action="demo_form.aspx">
   Enter a date before 1980-01-01:
   <input type="date" name="bday" max="1979-12-31">
   Enter a date after 2000-01-01:
   <input type="date" name="bday" min="2000-01-02">
   Quantity (between 1 and 5):
   <input type="number" name="quantity" min="1" max="5">
[...]
   <input type="range" name="voto" min="0" max="10" step="1">
   <input type="submit">
</form>
```

multiple New

In molti casi abbiamo bisogno che l'utente possa **inserire più valori per lo stesso input** (per esempio se gli stiamo chiedendo gli indirizzi e-mail di amici a cui inviare un invito).

Ecco che arriva in nostro soccorso l'attributo **multiple** che è un attributo booleano che funziona con i seguenti tipi di input: email e file.

Esempio

```
<form>
   <label>eMail a cui inviare l'invito:
      <input type="email" multiple name="friendEmail">
   </label>
   [...]
   <input type="reset" value="Cancella">
   <input type="submit" value="Invia">
</form>
```

type

Questo attributo viene usato per specificare un determinato tipo di input.

value

Questo attributo viene usato per assegnare un determinato valore al campo di input.

```
<input type="text" name="tuoTesto" value="inserisci qui il tuo testo" />
```

L'attributo value viene usato in modo diverso a seconda dei vari tipi di input:

- per i tipi "button", "reset" e "submit", indica il testo visualizzato sul pulsante;
- per i tipi "text", "password" e "hidden", indica il valore iniziale del campo di input;
- per i tipi "checkbox", "radio", "image", indica il valore associato al campo di input (ovvero il valore che viene memorizzato e processato alla pressione del pulsante di tipo submit)

Questo attributo è obbligatorio con <input type="checkbox"> e <input type="radio">.

placeholder New

Il valore dell'attributo **placeholder** è visualizzato all'interno di un input, o di una textarea, fin quando il campo è vuoto e non guadagna il focus (tramite il click o spostandosi su di esso il tasto Tab).

Semanticamente l'attributo **placeholder** dovrebbe essere valorizzato con valori accettabili dal form; dovrebbe contenere un esempio di ciò che l'utente andrà a scrivere nel campo.

Spesso il placeholder viene utilizzato **erroneamente** al posto della label descrivendo quindi cosa dovrebbero inserire.

Esempio

```
<form name="ricerca" method="post" action="/search">
   <label> Parola chiave:
      <input type="search" autocomplete="on" placeholder="article,
             section, ..." name="keyword" required maxlength="50">
   </label>
   <input type="submit" value="ricerca">
</form>
```

Parola chiave: article, section, ... ricerca

Figura 1 – Resa visiva di un input con l'attributo placeholder

L'attributo placeholder è applicabile ai seguenti tipi di input: text, search, url, tel, email, password.

readonly

È un attributo booleano che impedisce la modifica di un campo di input, **rendendolo accessibile solo in lettura**.

Esempio

```
<input value="leggere l'informativa" readonly />
```

leggere l'informativa

L'attributo `readonly` è applicabile ai seguenti campi di input: `input`, `textarea`, `select`, `radio`, `checkbox`, `password`.

required New

È un attributo booleano che serve a **rendere obbligatoria la compilazione dell'elemento a cui è applicato**. La condizione viene valutata al submit del form.

Esempio

```
<form name="commenti" method="post" action="">
   [...]
   <label>Messaggio:
      <textarea name="messaggio"
         placeholder="Scrivi qui il tuo messaggio (max 300 caratteri)"
         maxlength="300" required>
      </textarea>
   </label>
   [...]
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia il commento">
</form>
```

pattern New

In molti casi abbiamo bisogno di validare un determinato input verificando che il valore inserito sia conforme a determinate regole di creazione (per esempio potremmo volere che il campo password non contenga spazi). Possiamo contare in questi casi sull'attributo **`pattern`**.

Se viene indicato l'attributo pattern bisognerebbe indicare anche il **`title`** per dare una descrizione del formato richiesto, altrimenti il messaggio di errore sarà generico e probabilmente di poco aiuto.

Esempio

```
<form name="commenti" method="post" action="">
   [...]
   <label>Nick:
      <input type="text" name="nickname" autocomplete="on"
            required pattern="[a-z]{1}[a-z_]{2,19}"
            title="Un nickname è composto da lettere minuscole e '_';
            Sono consentiti da 3 a 20 caratteri."
            placeholder="your_nickname">
   </label>
   [...]
   <input type="reset" value="Cancella">
   <input type="submit" value="Invia">
</form>
```

In questo esempio si sta richiedendo che lo username sia una parola in minuscolo composta solo da lettere e da "_", di una lunghezza minima di 3 caratteri e massima di 20 e che non cominci con "_".

Il valore di `pattern`, se specificato, deve essere una **espressione regolare** valida.

Le espressioni regolari

Le espressioni regolari servono per trovare corrispondenze di **modelli** (**pattern**) su stringhe e costituiscono uno strumento molto utile.

I metacaratteri

Il modello più semplice è ovviamente quello costituito da una parola o, più in generale, da una sequenza di caratteri. È chiaro che in questo caso non è conveniente usare una delle funzioni per le espressioni regolari, perché queste sono molto "dispendiose" in termini di risorse impiegate, per cui il loro utilizzo dovrebbe essere limitato ai casi in cui non sia possibile utilizzare un'altra funzione (ad esempio, strpos(), o str_replace()).

Nella costruzione del modello, la cui corrispondenza andrà verificata, ci sono naturalmente una serie di regole da seguire, prima fra tutte quella che riguarda i cosiddetti "metacaratteri", ossia dei caratteri speciali, nel senso che, all'interno di un'espressione regolare, assumono un preciso significato. Questi caratteri sono:

- **.** che indica qualsiasi carattere (escluso un a capo)
- ***** che indica zero o più occorrenze (di un carattere o di un gruppo di caratteri)
- **?** che indica zero o una occorrenza (di un carattere o di un gruppo di caratteri)
- **{}** le parentesi graffe, che indicano il numero esatto, o minimo, o massimo, o l'intervallo di occorrenze (di un carattere o di un gruppo di caratteri)
- **+** che indica una o più occorrenze (di un carattere o di un gruppo di caratteri)
- **^** che indica l'inizio della stringa (o, se all'interno di una classe di caratteri, la negazione della stessa)
- **$** che indica la fine della stringa
- **|** che indica l'operatore OR
- **** il carattere di escape dei caratteri speciali
- **()** le parentesi tonde, destinate a contenere una sottostringa
- **[]** le parentesi quadre, destinate a contenere una "classe" di caratteri

Caratteristica comune a tutti questi metacaratteri è che se vogliono essere usati per il loro "valore letterale" (e non per quello speciale), devono essere preceduti da un backslash (\), per cui, ad esempio, l'esistenza di un punto interrogativo all'interno di una stringa, potrebbe essere accertata con questa espressione: \?.

Vediamo adesso di esaminare più nel dettaglio i metacaratteri, cominciando dalle varie parentesi. Le parentesi quadre [], come si è accennato, racchiudono una "classe di caratteri". Questo vuol dire che il modello può o deve contenere alcuni o tutti i caratteri in esse contenute; chiariamo il concetto con degli esempi:

Espressione	Descrizione
`[abc]`	questo modello è soddisfatto quando viene trovata una delle lettere, senza tener conto dell'ordine in cui sono presenti
`[^abc]`	questo modello è soddisfatto quando viene trovata una qualsiasi lettera diversa da quelle elencate all'interno delle parentesi
`[0-9]`	in questo modello è presente invece un intervallo di numeri, da 0 a 9; esso è soddisfatto quando viene trovato uno qualsiasi dei numeri compresi nell'intervallo
`[A-Z]`	in questo modello è presente un intervallo di caratteri (notare il segno – sta per "dalla A alla Z"), esso è soddisfatto quando viene trovato uno qualsiasi dei caratteri compresi nell'intervallo
`[a-z]`	in questo modello è presente un intervallo di caratteri, dalla a alla z; esso è soddisfatto quando viene trovato uno qualsiasi dei caratteri compresi nell'intervallo

`[A-z]`	in questo modello è presente un intervallo di caratteri, dalla A alla z; esso è soddisfatto quando viene trovato uno qualsiasi dei caratteri compresi nell'intervallo
`[a-z0-9\?]`	questo modello è leggermente più complesso, ma dovrebbe essere di facile comprensione. La corrispondenza viene trovata quando la stringa contiene una lettera (minuscola in questo caso), un numero o il carattere ? (notate il segno \ prima di ?, perché il punto interrogativo è un carattere speciale, che qui però assumiamo per il suo valore letterale);
`[^a-z]`	questo modello è soddisfatto quando viene trovato un qualsiasi carattere che non sia una lettera minuscola (notate il segno ^ che all'interno della classe, la nega). Naturalmente una classe di caratteri può essere seguita (e normalmente lo è) da uno dei metacaratteri che indicano il numero di volte in cui uno dei caratteri in essa contenuti, deve essere presente, riprendendo l'ultimo modello:
`[a-z0-9\?]?`	i caratteri contenuti nella classe devono essere presenti zero o una volta
`[a-z0-9\?]*`	i caratteri contenuti nella classe devono essere presenti zero o più volte
`[a-z0-9\?]{3}`	i caratteri contenuti nella classe devono essere presenti esattamente tre volte
`[a-z0-9\?]{1,3}`	i caratteri contenuti nella classe devono essere presenti da una a tre volte
`[a-z0-9\?]{3,}`	i caratteri contenuti nella classe devono essere presenti minimo tre volte
`[a-z0-9\?]{,3}`	i caratteri contenuti nella classe devono essere presenti massimo tre volte

Le parentesi

Le parentesi graffe indicano il numero esatto, minimo, massimo o l'intervallo di volte in cui una un'esatta sequenza o una classe di caratteri, devono essere presenti in una stringa:

- **`{3}`** esattamente 3 volte;
- **`{3,}`** minimo 3 volte;
- **`{,3}`** massimo 3 volte;
- **`{1,3}`** da 1 a 3 volte.

Le **parentesi tonde**, invece, fanno riferimento ad una sottostringa, o una parte di stringa se preferite, che viene assunta per il suo esatto valore letterale.

Quindi ad esempio (abc) si riferisce all'esatta sequenza di caratteri abc, a differenza, come abbiamo visto, di [abc] che si riferisce invece ad uno dei tre caratteri.

Ovviamente anche le parentesi tonde, possono essere usate con quei metacaratteri che indicano il numero di volte in cui la sottostringa deve ripetersi, per cui l'espressione (casa)? indica la presenza opzionale della parola casa (o che la parola deve essere presente zero o una volta).

Descriviamo adesso brevemente, gli altri metacaratteri. Partiamo dal **punto** che sta per qualsiasi carattere escluso un a capo, per cui, ad esempio, l'espressione (.)+ indica qualsiasi carattere ripetuto una o più volte (nella pratica è difficile che questo modello non trovi corrispondenza …).

Dei caratteri *****,**?** e **+** abbiamo già detto in relazioni alle classi e alle sottostringhe. Il carattere | (**pipe**) indica l'operatore **OR** e consente, quindi, di presentare più alternative per un'espressione; ad esempio (bello|normale|brutto) va letta come "bello o normale o brutto" ed è quindi soddisfatta quando solo una delle tre parole viene trovata nella stringa analizzata.

Il carattere **^** assume una duplice valenza, a seconda del punto in cui si trovi all'interno dell'Espressione Regolare ed occorre quindi porre molta attenzione nel suo uso.

Se posto all'inizio del modello, tale carattere indica l'inizio esatto dello stesso: ^(ciao) indica infatti che la stringa deve iniziare con la parola ciao. Ma l'accento circonflesso, se posto all'interno di una classe di caratteri, nega la stessa: [^0-9] indica qualsiasi carattere che non sia un numero.

Infine, il carattere **$** indica la fine di una stringa, per cui se viene usato in combinazione con ^, è possibile costruire un'Espressione Regolare che indichi un modello esattamente contenuto in una stringa, ad esempio ^Ciao come stai \?$, o che indichi l'esatto inizio e l'esatta fine di una stringa, ad esempio ^(Ciao) [a-zA-Z]+ (come stai \?)$.

Le Espressioni Regolari, conoscono anche, per così dire, delle abbreviazioni per ottenere ciò che si desidera, in relazione, ad esempio, alle classi di caratteri usate più di frequente. Eccone di seguito un breve schema riepilogativo:

- **\d** equivale a [0-9]
- **\D** equivale a [^0-9]
- **\w** equivale a [0-9A-Za-z]
- **\W** equivale a [^0-9A-Za-z]
- **\s** equivale a [\t\n\r]
- **\S** equivale a [^ \t\n\r]

Analogamente, esistono delle classi di caratteri predefinite:

[[:alpha:]] indica qualsiasi lettera, maiuscola o minuscola

[[:digit:]] indica qualsiasi cifra

[[:space:]] indica tutti i caratteri di spazio (\t\r\n)

[[:upper:]] indica le lettere maiuscole

[[:lower:]] indica le lettere minuscole

[[:punct:]] indica i caratteri di punteggiatura

[[:xdigit:]] indica i valori esadecimali

I valori dell'attributo type del tag input

Valore	HTML5	Descrizione
button		Crea un pulsante
checkbox		Crea una casella di selezione
color	New	Crea un widget per la selezione di un colore in una palette di colori
date	New	Crea un "datepicker" per la selezione della data
datetime	New	Crea un "datepicker" per la selezione di data, ora e fuso orario
datetime-local	New	Crea un "datepicker" per la selezione di data e ora
email	New	Crea un campo destinato all'inserimento di un indirizzo email
file		Crea un campo di testo e un pulsante "Sfoglia" per l'invio di file
hidden		Crea dei campi di input invisibili all'utente (nascosti)
image		Utilizza un'immagine come pulsante di tipo submit
month	New	Crea un "datepicker" per la selezione del mese
number	New	Crea un campo destinato all'inserimento di un numero
password		Utilizza il simbolo • per sostituire ogni carattere inserito
radio		Crea un pulsante "radio"
range	New	Crea un campo per inserire un numero tramite uno "slider"
reset		Crea un pulsante per azzerare/cancellare tutti i campi del form
search	New	Crea un campo di ricerca
submit		Crea un pulsante di invio
tel	New	Crea un campo adatto all'inserimento di numeri di telefono
text		Crea un campo di tipo testo
time	New	Crea un campo per la selezione dell'ora
url	New	Crea un campo destinato all'inserimento di un indirizzo web
week	New	Crea un "datepicker" per la selezione della settimana

checkbox

Con le **checkbox** possiamo consentire all'utente di operare delle scelte multiple. Ad esempio:

```
<form action="">
<fieldset>
   <legend>Linguaggi conosciuti</legend><br>
   <input type="checkbox" name="html" value="html"/> html
   <br />
   <input type="checkbox" name="css" value="css"/> css
   <br />
   <input type="checkbox" name="javascript" value="javascript"/>
   JavaScript
</fieldset>
</form>
```

che produce:

Linguaggi conosciuti

☐ html

☐ css

☐ JavaScript

Si possono anche selezionare uno o più valori che vengono spuntati di default, grazie all'attributo **checked**:

```
<form action="">
   <input name="html" type="checkbox" value="html" checked />
</form>
```

Ecco il risultato:

☑

Inoltre, grazie all'attibuto **disabled** possiamo rendere una casella non accessibile:

```
<form action="">
   <input name="html" type="checkbox" value="html" disabled />
</form>
```

radio

I **radio** button ("bottoni circolari") consentono di effettuare una scelta esclusiva. In questo caso quindi una scelta esclude l'altra. Per ottenere questo effetto **i campi devono avere lo stesso nome e differente valore**:

```
<form action="">
   <fieldset>
      <legend>Linguaggi conosciuti</legend>
      HTML <input type="radio" name="linguaggio" value="html"/>
      CSS <input type="radio" name="linguaggio" value="css"/>
      JavaScript <input type="radio" name="linguaggio"
                        value="javascript"/>
   </fieldset>
</form>
```

che viene così visualizzato:

Linguaggi conosciuti HTML ◉ CSS ○ JavaScript ○

Anche in questo caso è possibile assegnare un **valore di default** o **disabilitare un pulsante**.

```
<form action="">
   <input type="radio" name="linguaggio" value="html" checked disabled >
</form>
```

cioè:

submit

Non c'è form che si rispetti senza bottone di invio. La sintassi tradizionale per creare un **pulsante di invio** è:

```
<input type="submit" value="invia I dati">
```

Ad esempio:

```
<form action=http://www.html.it target="_blank">
   <input type="submit" value="visita HTML.it">
</form>
```

cioè:

visita HTML.it

reset

Un altro bottone utile è il **"reset"** che – una volta premuto – consente di riportare il form allo stato originario, cancellando ogni cosa scritta finora dall'utente. Ecco un esempio:

```
<form action="">
   <input type="text"><br>
   <input type="reset" value="Cancella">
</form>
```

Cancella

password

Il campo di tipo **password** mostra dei pallini (come questo •) al posto dei caratteri inseriti:

```
<input type="password" maxlength="8" size="8" value="mia_password" />
```

risultato:

Nota: la codifica fornisce una protezione soltanto per chi eventualmente stia sbirciando sul monitor dell'utente. L'invio dei dati attraverso il Web, se non vengono adottate altre misure di sicurezza, avviene "in chiaro".

file

Il tipo **"file"**, consente di inviare un file sul server, nel caso in cui la pagina di risposta sia stata programmata correttamente. La sintassi è:

```
<form action="">
   <input name="fileUtente" type="file" size="20" />
</form>
```

che dà come risultato:

hidden

Potremmo avere la necessità di passare dei parametri "di servizio", senza far percepire la loro presenza all'utente. In questo caso possiamo utilizzare dei campi nascosti, presenti all'interno del form, ma invisibili all'utente (ricordiamoci sempre di specificare la coppia **"nome-valore"**):

```
<input type="hidden" name="urlDiProvenienza" value="www.html.it">
```

tel New

È possibile utilizzare l'elemento input con `type="tel"` per creare un **campo adatto all'inserimento di numeri di telefono**.

A differenza degli input di tipo `email` e `url`, questo tipo **non impone un particolare formato**. Ciò è intenzionale. In pratica, i campi di numero di telefono sono campi liberi, perché, a livello intenzionale, i numeri possono essere scritti in diversi modi. È comunque possibile usare l'attributo `pattern` per forzare un determinato formato.

I **dispositivi mobili** possono presentare tastiere personalizzate per facilitare l'inserimento come mostrato nelle immagini che seguono (la prima è relativa a iPhone/iOS, mentre la seconda a un sistema Android).

Figura 1 – Tastiera iPhone

Figura 2 – Tastiera Android

Esempio

```
<form>
   <label>Inserisci il tuo numero di telefono:
      <input type="tel" name="myTelephone">
   </label>
   <input type="submit" value="Invia" >
</form>
```

Questo codice produce visivamente un normale `<input type="text">`.

search New

È possibile utilizzare l'elemento `input` con `type="search"` per creare un campo di ricerca. Questo campo è, ovviamente, un campo libero nel senso che non impone nessun pattern.

Nella maggior parte dei browser non c'è alcuna differenza tra un campo di tipo `text` e un campo `search`, ma in Safari su Mac OS X abbiamo un comportamento particolare:

1. visivamente l’input ha i bordi arrotondati;
2. se scriviamo qualcosa nel campo compare una piccola X sulla destra che, se cliccata, svuota il campo.

Figura 1

Esempio

```
<form name="ricerca" method="post" action="/search">
   <label> Parola chiave:
      <input type="search" autocomplete="on"
             placeholder="article, section, ..."
             name="keyword" required maxlength="50">
   </label>
   <input type="submit" value="Ricerca">
</form>
```

Come detto precedentemente, questo codice nella maggior parte dei browser produce visivamente un normale `<input type="text">`.

url New

Si usa l’elemento `input` con `type="url"` per creare **un campo destinato all’inserimento di un indirizzo web**.

Il tipo `url`, se specificato, dovrebbe rappresentare l’inserimento di un URL assoluto, ovvero nel formato `http://www.sito.com/etc`.... Nel caso in cui il valore inserito non sia valido, viene sollevata, nei browser che supportano il tipo `url`, un’eccezione che non riconosce il pattern.

I dispositivi mobili possono presentare tastiere personalizzate per facilitare l’inserimento. iPhone modifica la sua tastiera eliminando la barra spaziatrice e mettendo il punto, lo slash e l’estensione “.com” (Figura 1). Android, invece, visualizza attualmente la tastiera standard.

Figura 1 – Tastiera iPhone con un campo di tipo url

Esempio

```
<form name="commenti" method="post" action="">
   [...]
   <label> Www:
      <input type="url" name="url" autocomplete="on"
             placeholder="http://mywebsite.com">
   </label>
   [...]
   <input type="reset"  value="Resetta il form">
   <input type="submit" value="Invia il commento">
</form>
```

Produce visivamente un normale `<input type="text">`

email New

L'elemento input con type="**email**" viene usato per creare un **campo per inserire un indirizzo e-mail**.

L'input con tipo email, se specificato, dovrebbe rappresentare l'inserimento di indirizzi e-mail. Una fondamentale condizione di validità, dunque, sarà rappresentata dalla presenza del simbolo @. Nel caso in cui il valore inserito non sia valido viene sollevata un'eccezione.

I dispositivi mobili possono presentare, anche in questo caso, tastiere ad hoc. iPhone modifica la sua tastiera mostrando la chiocciola e il punto come visualizzato in figura 1. Android attualmente visualizza la tastiera standard.

Figura 1 – Tastiera iPhone con un campo di tipo email

Esempio

```
<form name="commenti" method="post" action="">
   [...]
   <label> Email:
      <input type="email" name="email" autocomplete="on"
             placeholder="email@domain.ext">
   </label>
   [...]
   <input type="reset"  value="Resetta il form">
   <input type="submit" value="Invia il commento">
</form>
```

Il codice visto produce visivamente un normale <input type="text">.

number New

È possibile utilizzare l'elemento input con type="**number**" per creare **un campo destinato all'inserimento di un numero**.

I dispositivi mobili possono presentare tastiere personalizzate per facilitare l'inserimento come mostrato nelle immagini che seguono (fanno riferimento a iPhone/iOS e Android).

Figura 1 – Tastiera iPhone con un input di tipo number

Figura 2 – Tastiera Android con un input di tipo number

Attributi specifici per il type number

HTML5 mette a disposizione un set di attributi specifici per i campi di tipo `number`. Servono a specificare delle limitazioni per il valore di questo attributo. Questi attributi sono `min`, `max` e `step`.

`min`

Specifica il minimo valore permesso. La sintassi è semplice: `min="1"` permette solo l'inserimento di numeri positivi. I valori permessi sono, ovviamente, numeri.

`max`

Specifica il massimo valore permesso. `max="10"` permette solo l'inserimento di numeri inferiori o uguali a 10. Il valore di questo attributo deve essere maggiore del valore dell'attributo `min` (se specificato).

`step`

L'attributo `step` indica la granulosità che deve avere il valore, limitando i valori permessi. Il valore di `step` se specificato deve essere un numero (anche non intero) maggiore di zero oppure la stringa "`any`" (che equivale a non inserire l'attributo).

La sintassi è anche in questo caso molto semplice: `step="3"` influenza i valori permettendo valori come -3, 0, 3, 6 ma non -1 o 2.

Un esempio potrebbe avere questa forma:

```
<form name="commenti" method="post" action="">
   [...]
   <label>Età:
      <input type="number" name="age" min="13" max="130" step="1">
   </label>
   [...]
   <input type="reset"  value="Resetta il form">
   <input type="submit" value="Invia il commento">
</form>
```

In questo caso stiamo chiedendo di inserire un'età compresa tra i 13 e i 130 anni (estremi compresi) e i valori accettati sono interi.

Nella maggior parte dei browser si produce attualmente un normale `<input type="text">`, ma nei browser che supportano `number` abbiamo:

Figura 1 – Un input di tipo number

range New

Molto simile semanticamente all'`input type="number"`, il `type="range"` permette agli utenti di **inserire un numero tramite uno slider**.

Attributi specifici per il type range

HTML5 mette a disposizione un set di attributi specifici per il tipo `range` (che sono gli stessi del `type="number"`): servono a specificare delle limitazioni per il valore di questo attributo. Questi attributi sono `min`, `max` e `step`.

- `min`: specifica il minimo valore permesso. Esempio: `min="1"`, che permette solo di passare numeri da 1 in su.
- `max`: specifica il massimo valore permesso. Esempio: `max="10"`, che permette solo di inviare numeri inferiori o uguali a 10. Il valore di questo attributo deve essere maggiore del valore dell'attributo `min` (se specificato).
- `step`: indica la granulosità che deve avere il `value` limitando i possibili valori passati. Il valore di `step` se specificato deve essere un numero (anche non intero) maggiore di zero oppure la stringa "any" (che equivale a non inserire l'attributo). Esempio: `step="3"`, che influenza i valori inseriti passando valori come -3, 0, 3, 6.

Esempio

```
<form name="commenti" method="post" action="">
   [...]
   <label>Voto:
      <input type="range" name="voto" min="0" max="5" step="1">
   </label>
   [...]
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia il commento">
</form>
```

Ecco come appare un input di tipo **`range`** sui browser che lo supportano:

Figura 1 – Un input di tipo range

color New

L'elemento `input` con `type="`**`color`**`"` dovrebbe creare un color picker, quel tipo particolare di widget utile per la selezione di un colore a partire da una palette di colori. Una volta selezionato il colore, il campo passa alla nostra pagina di ricezione un colore RGB esadecimale composto da 6 cifre.

Nella maggior parte dei browser non c'è alcuna differenza tra un campo di tipo text e un campo color, ma su Opera abbiamo un comportamento particolare:

1. visivamente abbiamo una select particolare con i colori (esempio in figura 1);
2. se clicchiamo abbiamo una scelta dei colori base (esempio in figura 2);
3. se clicchiamo su "Altro" richiamiamo il color picker di sistema (esempio su Opera su Mac OS X in figura 3).

Figura 1 – Input color su Opera: stato iniziale

Figura 2 – Input color su Opera: selezione del colore

Figura 3 – Input color su Opera: color-picker di sistema

Un input di tipo `color` si crea così:

```
<form>
   <label>Seleziona il colore di sfondo:
      <input type="color" name="mybackground">
   </label>
   <input type="reset"  value="Resetta la form">
   <input type="submit" value="Invia">
</form>
```

Nuovi tipi di input per la gestione delle date

Tutti i programmatori prima o poi hanno avuto a che fare con **la gestione delle date**, con tutti i problemi che ne conseguono. HTML5 mette a disposizione nuovi tipi di input concepiti per facilitare il compito dei programmatori nell'inserimento di date da parte degli utenti.

Dal momento che ci sono diversi tipi di date di cui potremmo aver bisogno, ecco che sono stati implementati nella specifica diversi tipologie. In particolare:

- **`datetime`**: gestisce sia la data che l'ora (con fuso orario);
- **`datetime-local`**: gestisce sia la data che l'ora (senza fuso orario);
- **`date`**: gestisce le date;
- **`month`**: gestisce i mesi;
- **`week`**: gestisce le settimane;
- **`time`**: gestisce l'ora.

Per tutti questi input abbiamo degli attributi specifici che sono:

- `min`: che rappresenta il minimo valore accettato;
- `max`: che rappresenta il massimo valore accettato;
- `step`: che rappresenta la granulosità dei valori accettati.

Vediamoli nel dettaglio.

datetime

Serve per permettere l'**inserimento di data e ora** in un solo colpo. Visivamente nei browser che lo supportano abbiamo la generazione di un **datepicker** in cui abbiamo la possibilità di selezionare un giorno e con l'opzione di mettere anche l'ora, come nell'immagine qui sotto. Ecco cosa avviene quando clicchiamo su quella che sembra essere una `select` (lo screenshot è di Opera):

Figura 1 – Input datetime su Opera

Nei sistemi che non supportato il tipo `datetime`, viene generato un normale input di testo.

Esempio

```
<form>
   <label>Specificare data e ora di prenotazione
      <input type="datetime" name="mydatetime">
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo datetime

- `min`: il valore di questo attributo deve essere una data e ora con il fuso orario valida;
- `max`: il valore di questo attributo deve essere una data e ora con il fuso orario valida e deve essere maggiore del valore dell'attributo `min` se specificato;
- `step`: il valore di questo attributo deve essere un intero e rappresenta i secondi. Il valore di default è di 60 secondi.

datetime-local

È del tutto simile a `datetime`, con l'unica differenza che non vengono passate informazioni sul fuso orario. Ecco come appare su Opera:

Figura 2 – Input datetime-local su Opera

Nei sistemi che non supportano il `datetime-local` viene generato un normale `input type="text"`.

Esempio

```
<form>
   <label>Specificare data e ora di prenotazione
      <input type="datetime-local" name="mydatetime">
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo datetime-local

- `min`: il valore di questo attributo deve essere una data e ora valida;
- `max`: il valore di questo attributo deve essere una data e ora valida e deve essere maggiore del valore dell'attributo `min` se specificato;
- `step`: il valore di questo attributo deve essere espresso in secondi. Il valore di default è di 60 secondi.

date

Serve per inserire una data. Nei browser che lo supportano si ottiene un "datepicker" in cui abbiamo la possibilità di selezionare un giorno:

Figura 1 – Input date su Opera

Nei sistemi che non supportato il tipo `date` viene generato un normale campo di testo.

Esempio

```
<form>
   <label>Specificare la data di prenotazione
      <input type="date" name="mydatetime">
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo date

- `min`: il valore di questo attributo deve essere una data valida;
- `max`: il valore di questo attributo deve essere una data valida e deve essere maggiore del valore dell'attributo `min` se specificato;
- `step`: il valore di questo attributo deve essere espresso in giorni. Il valore di default è di 1 giorno.

month

Serve per permettere di selezionare un mese dell'anno. In figura 2 il "datepicker" per selezionare un mese generato su Opera:

Figura 2 – Input month su Opera

Nei sistemi che non supportato `month` viene generato un normale campo di testo.

Esempio

```
<form>
   <label>Specificare il mese di prenotazione
      <input type="month" name="mydatetime">
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo month

- `min`: il valore di questo attributo deve essere un mese valido;
- `max`: il valore di questo attributo deve essere una mese valido e deve essere maggiore del valore dell'attributo `min` se specificato;
- `step`: il valore di questo attributo deve essere espresso in mesi. Il valore di default è di 1 mese.

week

Viene usato per la **selezione di una determinata settimana dell'anno** (anno–numero di settimana). In fig. 5 il widget prodotto dall'inserimento di questo tipo di input su Opera:

Figura 5 – Input week su Opera

Nei sistemi che non supportato `week` viene creato un normale campo di testo.

Esempio

```
<form>
   <label>Specificare la settimana di prenotazione
      <input type="week" name="mydatetime">
   </label>
   <input type="reset" value="Resetta il form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo week

- `min`: il valore di questo attributo deve essere una settimana valida.
- `max`: il valore di questo attributo deve essere una settimana valida e deve essere maggiore del valore dell'attributo `min` se specificato.
- `step`: il valore di questo attributo deve essere espressa in settimane. Il valore di default è di 1 settimana.

time

Serve per **selezionare e inserire una determinata ora del giorno**. Ancora una schermata da Opera:

Figura 1 – Input time su Opera

Nei sistemi che non supportato il `time` genera un normale `input type="text"`.

Esempio

```
<form>
   <label>Specificare l'ora di prenotazione
      <input type="time" name="mydatetime">
   </label>
   <input type="reset" value="Resetta la form">
   <input type="submit" value="Invia">
</form>
```

Gli attributi specifici per il tipo time

- `min`: il valore di questo attributo deve essere una ora valida;
- `max`: il valore di questo attributo deve essere una ora valida e deve essere maggiore del valore dell'attributo min se specificato;
- `step`: il valore di questo attributo deve essere espresso in secondi. Il valore di default è di 60 secondi.

<textarea>

Esempio

```
<textarea name="testo" rows="5" cols="40">
  qui puoi scrivere il tuo testo
</textarea>
```

Se si ha la necessità di indicare un campo che consenta di inserire una grande quantità di testo conviene utilizzare una **`textarea`** (area di testo). Ecco il risultato:

L'attributo **`rows`** indica il numero di righe della `textarea`, **`cols`** il numero di caratteri (cioè di colonne) che ogni riga può contenere. Le dimensioni dell'area di input possono essere specificate con le proprietà `width` e `height` definite in un foglio di stile (CSS).

Le impostazioni predefinite prevedono la visualizzazione del testo con un numero illimitato di caratteri di tipo Courier.

Come si può vedere, se si vuol indicare del testo predefinito in questo caso bisogna inserirlo fra l'apertura e la chiusura del tag.

Gli attributi di <textarea>

Gli attributi di questo tag sono gli stessi che abbiamo già esaminato per l'elemento <input>:

`autofocus`, `disabled`, `form`, `maxlength`, `name`, `placeholder`, `readonly`, `required`.

<select>

Esempio

```
<select>
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
   <option value="mercedes">Mercedes</option>
   <option value="audi">Audi</option>
</select>
```

Grazie al tag **`<select>`** è possibile costruire dei menu di opzioni. In questo caso ciascuna voce deve essere compresa all'interno del tag **`<option>`** (la chiusura del tag è opzionale) e il valore deve essere specificato attraverso l'attributo **"`value`"**. Con l'attributo **"`selected`"** si può indicare una scelta predefinita:

```
<form>
   <fieldset>
      <legend>Siti per webmaster</legend>
      <select name="siti" >
         <option value="http://www.html.it"
                        selected>www.html.it</option>
         <option value="http://freephp.html.it">frephp.html.it</option>
         <option value="http://freasp.html.it">freasp.html.it</option>
      </select>
   </fieldset>
</form>
```

che dà luogo a:

Siti per webmaster

www.html.it ▾

Siccome i menu di scelta tendono a diventare particolarmente lunghi, nell'HTML 4 è stato introdotto il tag **`<optgroup>`** che consente di suddividere le varie possibilità di scelta in gruppi tramite l'utilizzo di apposite etichette (attributo **`label`**). Ecco l'esempio:

```
<form action="">
   <select name="siti">
      <optgroup label="siti per webmaster">
         <option value="http://www.html.it">HTML.it</option>
         <option value="http://www.html.it/javascript">JavaScript
                        HTML.it</option>
         <option value="http://www.html.it/css">CSS HTML.it</option>
      </optgroup>
      <optgroup label="risorse per webmaster">
         <option value="http://font.html.it">font.html.it</option>
         <option value="http://cgipoint.html.it">
                        cgipoint.html.it</option>
      </optgroup>
   </select>
</form>
```

che dà luogo al seguente menu:

Infine con il tag **select** è possibile impostare anche delle scelte multiple. Come si può vedere, utilizzando l'attributo **"multiple"**, l'aspetto del tag **select** cambia notevolmente:

```
<form action="">
   <label>Quale sito visiti?<br />
      <select name="siti" multiple>
         <option value="http://www.html.it">www.html.it</option>
         <option value="http://freephp.html.it">frephp.html.it</option>
         <option value="http://freasp.html.it">freasp.html.it</option>
         <option value="http://font.html.it">font.html.it</option>
         <option value="http://cgipoint.html.it">cgipoint.html.it
         </option>
      </select>
   </label>
</form>
```

cioè:

Utilizzando il tasto **CTRL** l'utente può così effettuare delle **scelte multiple**.

Tramite l'attributo **"size"** si può specificare il numero delle voci che devono comparire nel menu, e conseguentemente regolare l'altezza del menu, aggiungendo o togliendo la barra di scorrimento verticale.

```
<form action="">
   <label>Quale siti visiti?<br>
   <select name="siti" size="5" multiple>
      <option value="http://www.html.it">www.html.it</option>
      <option value="http://freephp.html.it">frephp.html.it</option>
      <option value="http://freasp.html.it">freasp.html.it</option>
      <option value="http://font.html.it">font.html.it</option>
      <option value="http://cgipoint.html.it">cgipoint.html.it
      </option>
   </select>
   </label>
</form>
```

che viene così visualizzato:

Gli attributi di <select>

Oltre a **multiple** e **size**, gli attributi del tag **select** sono gli stessi che abbiamo già esaminato per l'elemento <input>:

autofocus, disabled, form, name, required

<fieldset>

Esempio

```
<form>
   <fieldset>
      <legend>Dati personali</legend>
      Cognome: <input type="text"><br />
      Nome: <input type="text"><br />
      Email: <input type="email"><br />
      Data di nascita: <input type="date">
   </fieldset>
</form>
```

Per la loro natura di "raccoglitori di informazioni", i moduli tendono a ingigantirsi e diventare lunghissimi. Per questo, con HTML 4 sono stati introdotti dei tag per fare un po' d'ordine all'interno dei form.

Grazie al tag **<fieldset>** possiamo creare delle macro-aree all'interno dei form, e grazie al tag **<legend>**, possiamo indicare il nome di ciascuna macro-area.

Poniamo ad esempio di dover raccogliere i dati di un utente, raccogliendo dati anagrafici, residenza, domicilio e reperibilità sul lavoro.

Possiamo farlo con la seguente sintassi:

```
<form action="">
   <fieldset>
      <legend>Dati anagrafici</legend>
      <br /><br /><br />
   </fieldset>
   <fieldset>
      <legend>Residenza</legend>
      <br /><br /><br />
   </fieldset>
</form>
```

che dà come risultato:

come si può vedere vengono creati dei riquadri con un indicazione del tipo di contenuto.

<label>

Esempio

```
<form action="demo_form.asp">
   <label for="male">Male</label>
   <input type="radio" name="sex" id="male" value="male"><br>
   <label for="female">Female</label>
   <input type="radio" name="sex" id="female" value="female"><br>
   <input type="submit" value="Submit">
</form>
```

Un altro tag particolarmente utile è il tag **<label>**, che permette di indicare un'etichetta per il nome del campo. Esistono due metodi HTML per contrassegnare un campo di un form con un'etichetta.

Il primo è usare l'attributo "**for**" sulla **label**, con un valore corrispondente all'**id** dell'elemento associato:

```
<label for="nome">Nome utente:</label>
<input type="text" id="nome" />
```

Il secondo è inserire il campo del form all'interno della **label**:

```
<label>Nome utente: <input type="text" /></label>
```

Esempio

```
<form action="">
   <fieldset>
      <legend>Dati anagrafici</legend>
      <label>Cognome e nome: <input type="text"></label>
   </fieldset>
</form>
```

che dà come risultato:

oppure (cambiando la posizione del testo):

```
<fieldset>
   <legend>Dati anagrafici</legend>
   <label><input type=”text”>: cognome e nome</label>
</fieldset>
```

che dà come risultato:

Come si può vedere il campo su cui si vogliono dare delle indicazioni deve essere compreso all’interno del tag **label** stesso.

Fonti: http://www.html.it – “Informatica e reti per i sistemi informativi aziendali” (Ed. Atlas) – “HTM5 espresso con CSS3 e ECMAScript5” (Ed. HOEPLI Informatica) – http://www.mrwebmaster.it – http://www.w3schools.com